Cryptology with CrypTool

Version 1.4.00

Introduction to Cryptography and Cryptanalysis
Scope, Technology and Future of CrypTool

www.cryptool.de
www.cryptool.com
www.cryptool.org

Bernhard Esslinger and CrypTool team, 2006

Content

Content

The CrypTool Project

- Origin in awareness program of a bank (in-firm training)
  → **Awareness for employees**
- Developed in Cooperation with Universities (improving education)
  → **media didactic**

1998 – **Project start** – effort more than 15 man-years

2000 – CrypTool available as **freeware**

2002 – CrypTool on **Bürger-CD-ROM from BSI** „Ins Internet – mit Sicherheit“

2003 – CrypTool becomes **Open-Source** – Hosting through University of Darmstadt (Prof. Eckert)

2004 – Awards

TeleTrusT	(TTT Förderpreis 2004)
NRW	(IT Security Award NRW)
RSA Europe	(Finalist of European Information Security Award 2004)

- **Developers**
  – Developed by People from different Companies and Universities
  → Additional project members or usable sources are always appreciated (up to now there are around 30 people working on CrypTool world wide).

Relevance of Cryptography

Typical Scenarios for using Cryptography in the daily life

Examples for Cryptography Usage

- Phone cards, cell phones, remote controls
- Cash machines, money transfer between banks
- Electronic cash, online banking, secure eMail
- Satellite TV, Pay TV
- Immobiliser systems in cars
- Digital Rights Management (DRM)

- Cryptography is no longer limited to agents, diplomats or the military. Cryptography is a modern, mathematically characterised science.
- Breakthrough for cryptography started with the broad use of the Internet
- For companies and governments it is important that systems are secure and …

  … users (clients, employees) have a certain understanding and awareness for IT security!

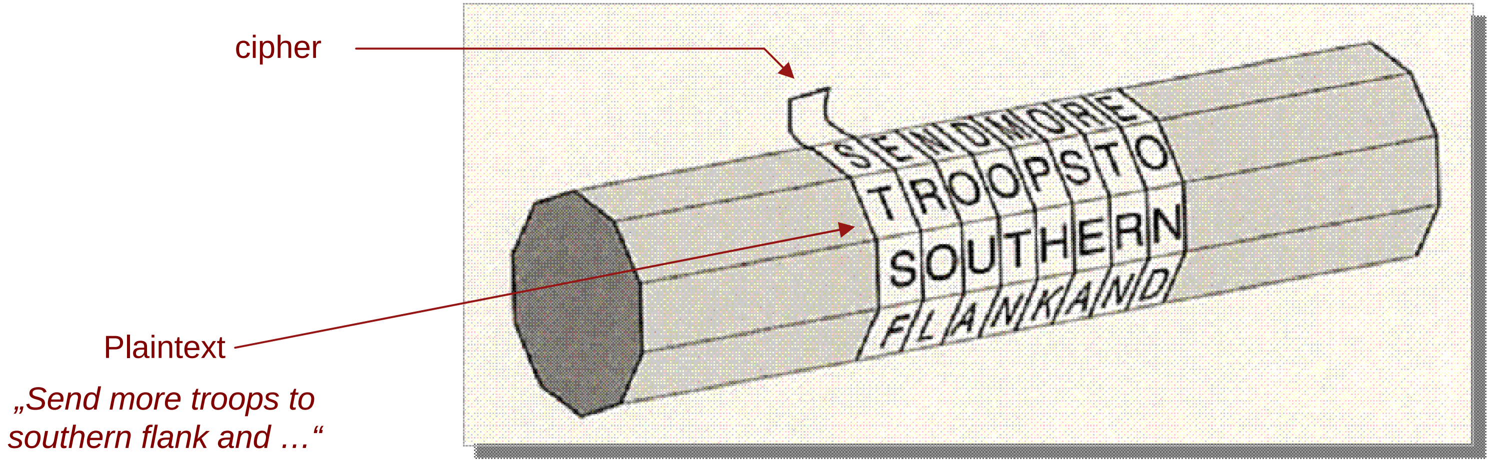
Cryptography – Objectives

Protection goals related to Cryptography

- **Confidentiality**
  - *Information is can practically not made available or disclosed to unauthorized individuals, entities or processes.*
- **Authentication**
  - *Authentication ensures that users are identified and those identities appropriately verified.*
- **Integrity**
  - *Integrity ensures that data has not been altered or destroyed in an unauthorized manner.*
- **Non-Repudiation**
  - *The principle that, afterwards, it can be proven that the participants of a transaction did really authorize the transaction and that they have no means to deny their participation.*

Examples of early cryptography (I)
Ancient encryption methods

- **Tattoo on a slave's head concealed by re-grown hair**
- **Atbash** (around 600 B.C.)
  - Hebrew secret language, reversed alphabet
- **Skytale from Sparta** (500 B.C.)
  - Described by Greek historian/author Plutarch (45 - 125 AD)
  - Two cylinders (wooden rod) with identical diameter
  - Transposition (plaintext characters are re-sorted)



Examples of early cryptography (II)

Symmetric Caesar encryption

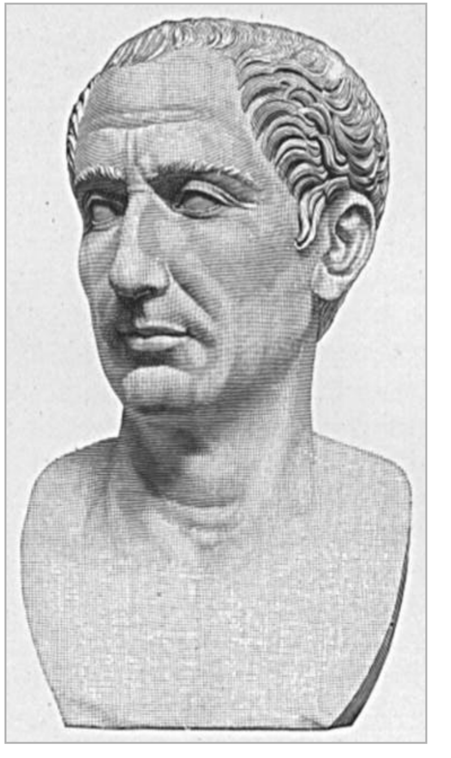
- **Caesar encryption** (Julius Cäsar, 100 - 44 v.Chr.)
- Simple substitution cipher

```
GALLIA  EST  OMNIS  DIVISA ...

Plaintext:        ABCDEFGHIJKLMNOPQRSTUVWXYZ
Secret alphabet:  DEFGHIJKLMNOPQRSTUVWXYZABC

JDOOLD  HVW  RPQLV  GLYLVD ...
```

- **Attack:** Frequency-analysis (typical character allocation)



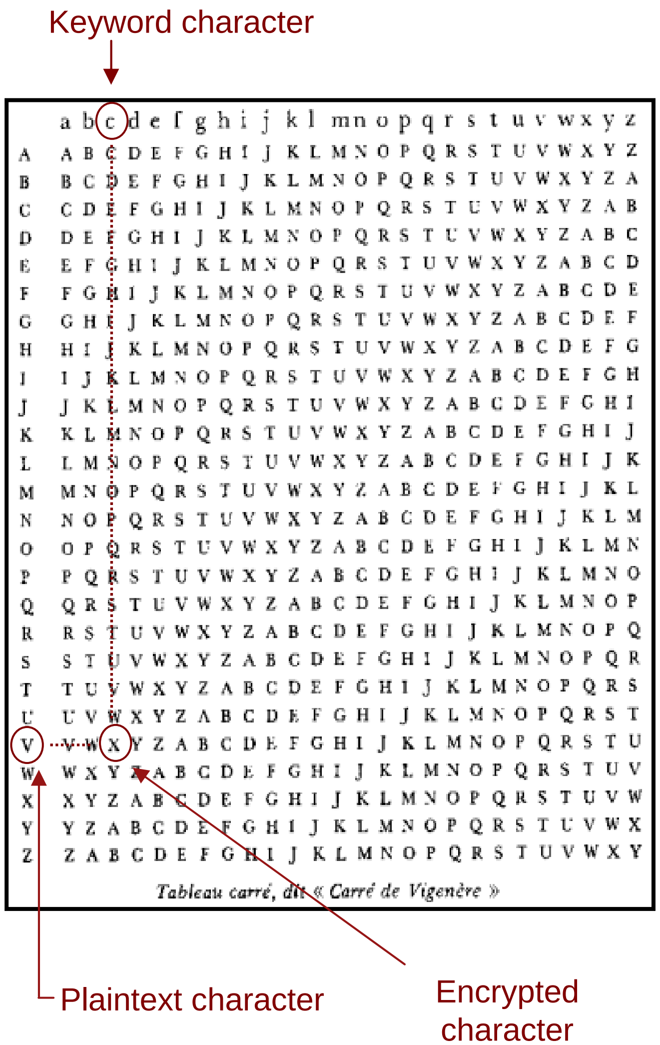
Presentation with CrypTool via the following menus:

- Animation: “Indiv. Procedures“ \ “Visualization of algorithms using ANIMAL“ \ “Caesar …“
- Implementation: “Crypt/Decrypt” \ “Symmetric (classic)” \ “Caesar / Rot 13”

Examples of early cryptography (II)

Symmetric Vigenère encryption

- **Vigenère Encryption** (Blaise de Vigenère, 1523-1596 AD)
- Encryption with a key word using a key table
- Key word: **`CHIFFRE`**
- Encrypting: **`VIGENERE`** becomes **`XPOJSVVG`**
- The plaintext character (V) is replace by the character in the corresponding row and in the column of the first key word character (C). The next plaintext character (I) is replaced by the character in the corresponding row and in the column of the next key word character (H), and so on.
- If all characters of the key word have been used, then the next key word character is the first key character.
- **Attack** (via Kasiski test): Plaintext combinations with an identical cipher text combination can occur. The distance of these patterns can be used to determine the length of the keyword. An additional frequency analysis can then be used to determine the key.



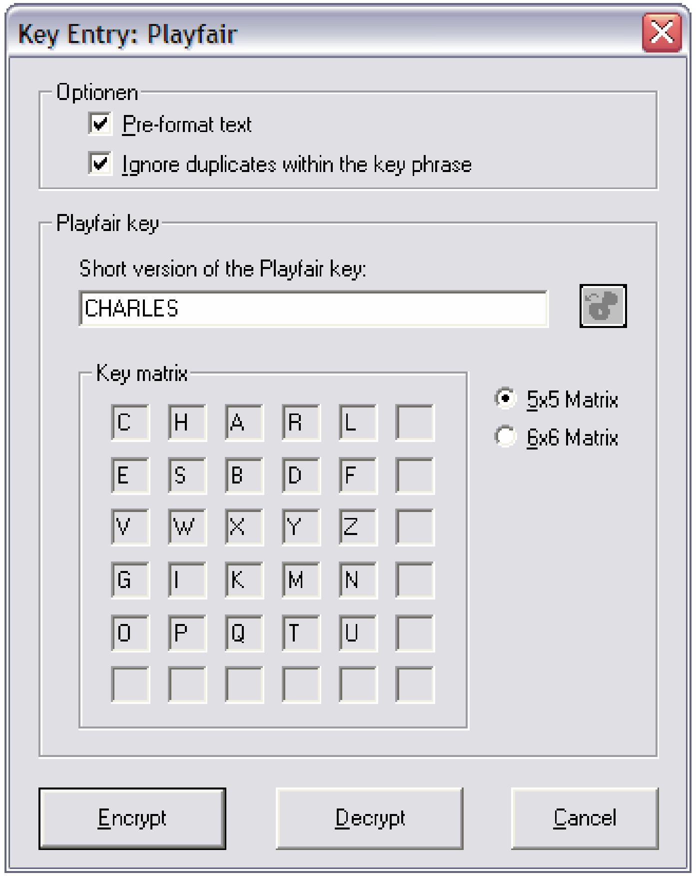
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tableau carré, dit « Carré de Vigenère »

Examples of early cryptography (IV)

Other symmetric encryption methods

- **Homophone substitution**
- **Playfair** (1854 Sir Charles Wheatstone, 1802-1875)
  - published by Baron Lyon Playfair
  - Substitution of a character pair by another based on a square-based alphabet array
- **Transfer of book pages**
  - Adaptation of the One-Time-Pad (OTP)
- **Hole template** (Fleißner)
- **Permutation encryption** („Double Dice")
  - Transposition / very effective



Cryptography in Modern Times

Cryptography developments in the last 100 years

Classic Methods

- … are still in use today.
  (since, not all can be done by a computer…)
- … and their principals of transposition and substitution are inputs for the design of modern algorithms:
  combinations of simple operations (a type of multiple encryption, a so called cascades of ciphers), on bit level, block cipher, rounds.

Encryption becomes

- … more sophisticated,
- mechanised or computerised and
- remains symmetric.

Examples of the first half of the 20th century

Mechanical encryption machines (rotor machines)

- **Enigma Encryption** (Arthur Scherbius, 1878-1929)
- More than 200.000 machines have been used in WW2
- The rotating cylinder set causes, that every character of the text becomes encrypted with a new permutation.
- Broken by massive effort of cryptography experts (around 7.000 person in UK) with decryption machines, captured original Enigmas and by intercepting daily status reports (e.g. weather reports).
- **Consequences of this successful crypto analysis**:

  "In general the successful crypto analysis of the engima encryption has been a strategic advantage, that has played a significant role in winning the war. Some historians assume that the break of the enigma code has shorten the war by several months or even a year."

  (translated from http://de.wikipedia.org/wiki/Enigma_%28Machine%29 - 6 March 2006)

Cryptography – Important Insights (I)

- **Kerckhoffs principle** (1883)
  - Separation of algorithm (method) and key

    Algorithm: “Shift alphabet by a certain number of positions to the left”
    Key: the “certain number of positions” (Caesar for example)
  - Kerckhoffs principle: The secret lies within the key and not within the algorithm or „No security through obscurity“
- **One Time Pad – Shannon / Vernam**
  - Demonstrably theoretically secure, but not usable in reality (only red phone)
- **Shannons concepts: Confusion and Diffusion**
  – Relation between M, C and K has to be as complex as possible
  – Every cipher text character should depend on as many plaintext characters and as many character of encryption key as possible
  – „Avalanche effect“ (small modification, big impact)
- **Trapdoor function** (one-way function)
  – Fast in one direction, very slow in the opposite direction
  – Only the secret key grants access to trapdoor

Examples for a breach of the Kerckhoffs principle

Secret lies within the key and not within the algorithm

- **Cell phone encryption penetrated** (December 1999)

  „Israeli researchers have discovered design flaws that allow the descrambling of supposedly private conversations carried by hundreds of millions of wireless phones. Alex Biryukov and Adi Shamir describe in a paper to be published this week how a PC with 128 MB RAM and large hard drives can penetrate the security of a phone call or data transmission in less than one second. The flawed algorithm appears in digital GSM phones made by companies such as Motorola, Ericsson, and Siemens, and used by well over 100 million customers in Europe and the United States.” […]

  *“Previously the GSM encryption algorithms have come under fire for **being developed in secret away from public scrutiny** -- but most experts say high security can only come from published code. Moran said "it wasn't the attitude at the time to publish algorithms" when the A5 ciphers was developed in 1989, but **current ones being created will be published for peer review**.”*

  [http://wired.lycos.com/news/politics/0,1283,32900,00.html]

Sample of a One Time Pad Adaptation

Clothes hanger of a Stasi agent
with a secret One Time Pad
(taken from: *Spiegel Spezial 1/1990)*

Key Distribution Problem

Key distribution for a symmetric encryption

If **2 persons** communicate with each other using symmetric encryption, they **need one secret key**.

If n persons communicate with each other, then they need $\mathbf{S_n = n(n-1) / 2}$ keys.

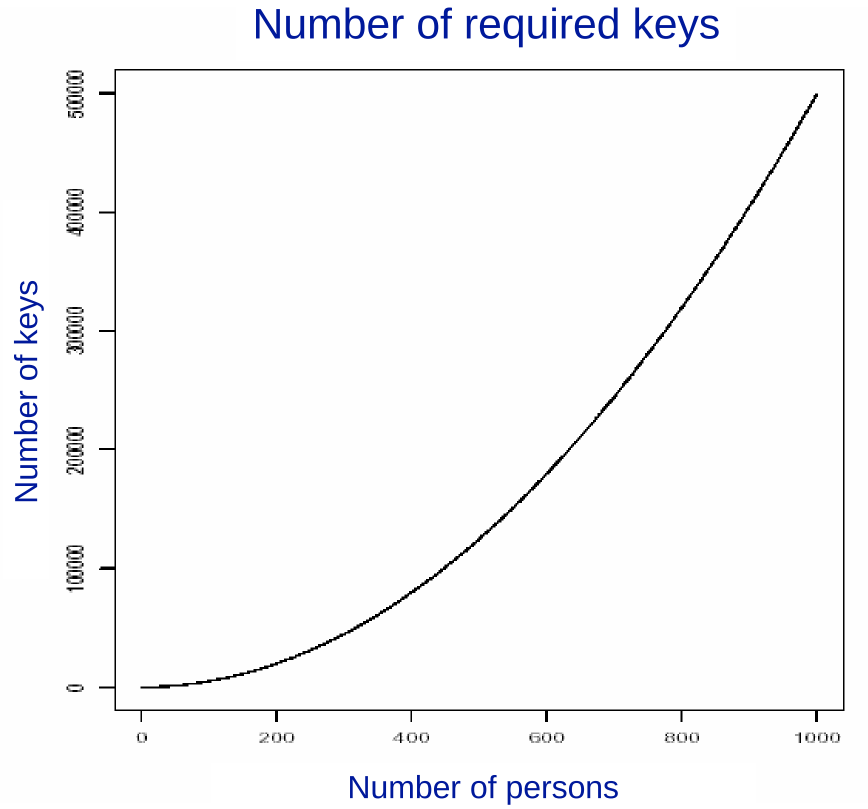
This means

n = 100 persons require

$\mathbf{S_{100} = 4.950}$ keys, and

n = 1.000 persons require

$\mathbf{S_{1000} = 499.500}$ keys.



Cryptography – Important Insights (II)

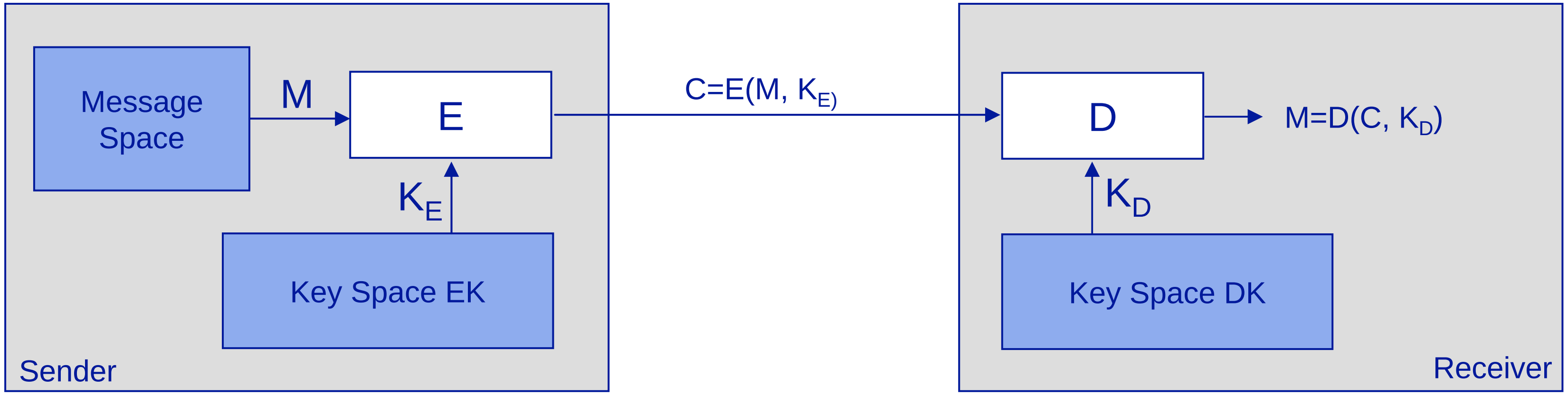
Solving the key distribution problem through asymmetric cryptography

- **Asymmetric cryptography**
  - For centuries it was believed that: Sender and receiver need same secret.
  - New: Every member needs a key pair (Solution of the key distribution problem)
  - **Asymmetric encryption**
    - „Everyone can lock a padlock or can drop a letter in a mail box."
    - MIT, 1977: Leonard Adleman, Ron Rivest, Adi Shamir (well known as RSA)
    - GCHQ Cheltenham, 1973: James Ellis, Clifford Cocks (admitted in public December 1997)
  - **Key distribution**
    - Stanford, 1976: Whitfield Diffie, Martin Hellman, Ralph Merkle (Diffie-Hellman Key Exchange)
    - GCHQ Cheltenham, 1975: Malcolm Williamson

Security in open networks (such as the internet) would be extremely expensive and complex without asymmetric cryptography!

Encryption and Decryption

Symmetric und asymmetric encryption



secret

a) Symmetric Encryption: $\mathbf{K_E = K_D}$ (e.g. DES)

b) Asymmetric Encryption: $\mathbf{K_E \neq K_D}$ (e.g. RSA)

public private/secret

Cryptography – Important Insights (III)

Increasing relevance of mathematics and information technology

- **Modern cryptography** is based on **mathematics**
  - Still new symmetric encryption methods such as AES (better performance and shorter key length compared to the asymmetric methods purely based on mathematical problems).
- The security of encryption methods heavily depends on the current status of **mathematics** and **information technology** (IT)
  - Computation complexity (meaning processing effort in relation to key length, storage demand and data complexity)
    -> see RSA: Bernstein, TWIRL-Device, RSA-160
  - Very high activity in current research:

    Factorisation, non-parallelize algorithm (because of quantum computing), better understanding of protocol weaknesses and random generators, ...).
- Serious mistake: *"Real mathematics has no effects on the war."* (G.H. Hardy, 1940)
- Vendors discover **security** as an essential **purchase criterion**

Demo in CrypTool

- *Statistical Analysis*

- *Encrypting twice is not always better:*

  Caesar: C + D = G (3 + 4 = 7)

  Vigenère: - CAT + DOG = FOZ [(2,0,19)+(3,14,6)=(5,14,25)]

  - "Hund" + "Katze" ="RUGCLENWGYXDATRNHNMH")

- *Vernam (OTP)*

- *AES (output key, brute-force analysis)*

Content

CrypTool and Cryptography – Overview

CrypTool features

Examples

Project / Outlook / Contact

CrypTool Features

E-Learning

1. What is CrypTool?

- Freeware program with graphical interface
- Cryptographic methods can be applied and analysed
- Comprehensive online help (understandable without deeper cryptography knowledge)
- Contains nearly all state-of-the-art cryptography functions
- Easy entry into modern and classical cryptography
- Not a *“hacker tool”*

2. Why CrypTool?

- Origin in awareness initiative of a commercial bank
- Developed in close cooperation with universities
- Improvement of university education and in-firm training

3. Target Group

- *Core group:* Students of computer science, business computing and mathematics
- *But also for:* computer users, application developers, employees
- *Prerequisite*: PC knowledge
- *Preferable:* Interest in mathematics and/or programming

Content of the program package

CrypTool Program

- All functions integrated in a *single* program with consistent graphical interface
- Runs on Win32
- Cryptography libraries from Secude and OpenSSL
- Long integer arithmetic from Miracl and GMP, Lattice base reduction via NTL (Shoup)

AES-Tool

- Standalone program for AES encryption (self extracting)

Educational game

- „Numbers Shark“ encourages the understanding of factors and prime numbers.

Comprehensive Online Help (HTML-Help)

- Context-sensitive help available via F1 for *all* program functions (including menus)
- Detailed use cases for a lot of program functions (tutorial)

Script (.pdf file) with background information

- Encryption methods • Prime factorisation • Digital signature
- Elliptic curves • public key certification • Basic number theory

Two short stories related to cryptography by Dr. C. Elsner

- „The Dialogue of the Sisters“ (a RSA variant as key element)
- „The Chinese Labyrinth“ (Numbers theory tasks for Marco Polo)

Features (I)

Cryptography

Classical cryptography

- Caesar
- Vigenère
- Hill
- Homophone Substitution
- Playfair
- ADFGVX
- Addition
- XOR
- Vernam
- Permutation
- Solitaire

Several options to easily understand the cryptography methods

- Selectable alphabet
- Options: handling of blanks, etc.

Crypto analysis

Attack on classical methods

- Cipher text Only
  - Caesar
  - Vigenère
  - Addition
  - XOR
  - Substitution
  - Playfair
- Known Plaintext
  - Hill
- Manual
  - Mono alphabetical substitution
  - Playfair
  - Solitaire

Supported analysis methods

- Entropy, floating frequency
- Histogram, N-Gram-Analysis
- Autocorrelation
- Periodicity

Features (II)

Cryptography

Modern symmetric Encryption

- IDEA, RC2, RC4, RC6, DES, 3DES
- Serpent, Twofish
- AES candidates of the last selection round
- AES (=Rijndael)

Asymmetric Encryption

- RSA with X.509-Certificats
- RSA-Demo
  - Understanding of examples
  - Selectable alphabet and block length

Hybrid Encryption (RSA + AES)

- Inter active data flow diagram

Crypto analysis

Brute-force-Attack on symmetric algorithm

- For all algorithms
- Assumption: Entropy of plaintext is small

Attack on RSA encryption

- Factorisation of RSA-module
- Lattice-Based attacks

Attack on hybrid encryption

- Attack on RSA or
- Attack on AES (side-channel attack)

Features (III)

Cryptography

Digital signature

- RSA with X.509-Certificates
  - Signature as interactive data flow diagram
- DSA with X.509-Certificates
- Elliptic Curve DSA, Nyberg-Rueppel

Hash functions

- MD2, MD4, MD5
- SHA, SHA-1, RIPEMD-160

Random generators

- Secude
- x^2 mod n
- Linear congruence generator (LCG)
- Inverse congruence generator (ICG)

Crypto analysis

Attack on RSA signature

- Factorisation of the RSA-module
- feasible up to 250 bits or 75 decimal places (on standard desktop PCs)

Attack on Hash functions / digital signature

- Generate Hash collisions for ASCII based text (birthday paradox) (up to 40 bit in around 5 min)

Analysis of random data

- FIPS-PUB-140-1 Test-Battery
- Periodicity, Vitany, Entropy
- Floating frequency, histogram
- n-Gram-Analysis, autocorrelation
- ZIP compression test

Features (IV)

Animation / Demos

- Caesar, Vigenère, Nihilist, DES with ANIMAL
- Hybrid encryption and decryption
- Generation and verification of digital signatures
- Diffie-Hellman key exchange
- Secret Sharing (with CRT or Shamir)
- Challenge-Response method (authentication)
- Side channel attack
- Graphical 3D presentation of (random) data streams
- Sensitivity of hash functions regarding plaintext modifications
- Numbers theory and RSA crypto system

Features (V)

Additional Functions

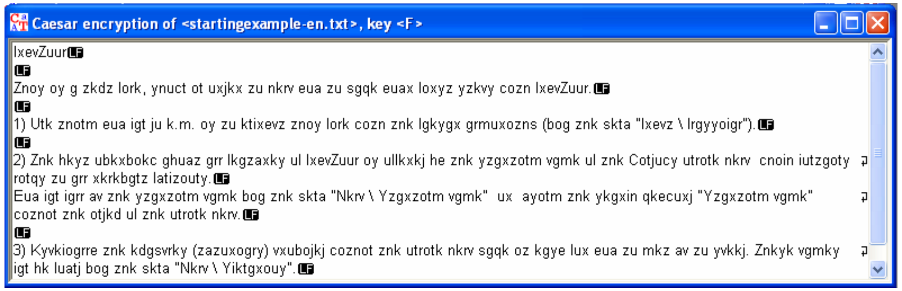
- Homophone and permutation encryption
- PKCS #12 import and export for PSEs (Personal Security Environment)
- Generate hashes of large files, without loading them
- Brute force attacks on symmetric algorithms
- …

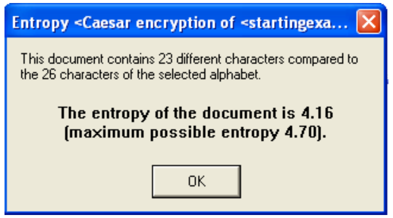
Language structure analysis

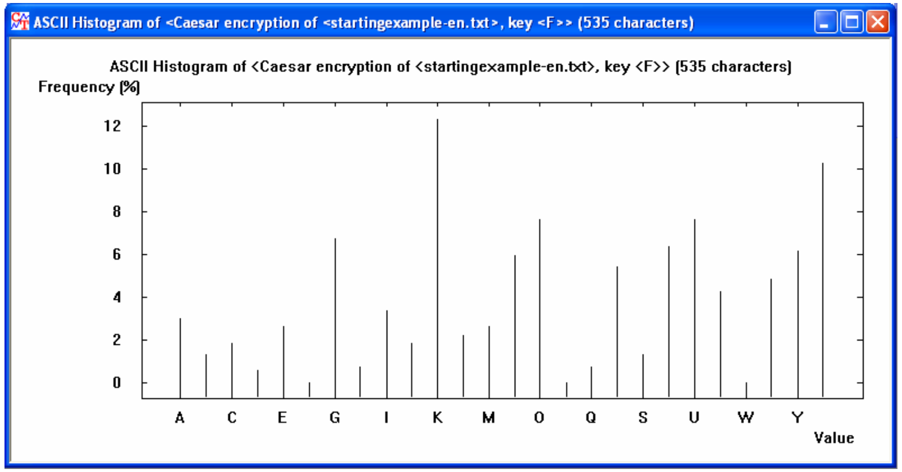
Language analysis options available in CrypTool

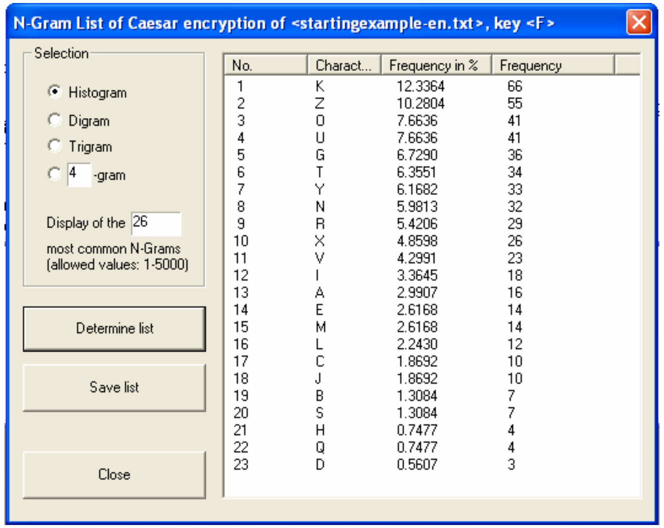
Number of characters, n-Gram, Entropy

- e.g. using CrypTool „Analysis / Tools for Analysis /...“









No.	Charact...	Frequency in %	Frequency
1	K	12.3364	66
2	Z	10.2804	55
3	O	7.6636	41
4	U	7.6636	41
5	G	6.7290	36
6	T	6.3551	34
7	Y	6.1682	33
8	N	5.9813	32
9	R	5.4206	29
10	X	4.8598	26
11	V	4.2991	23
12	I	3.3645	18
13	A	2.9907	16
14	E	2.6168	14
15	M	2.6168	14
16	L	2.2430	12
17	C	1.8692	10
18	J	1.8692	10
19	B	1.3084	7
20	S	1.3084	7
21	H	0.7477	4
22	Q	0.7477	4
23	D	0.5607	3

Demonstration of Interactivity (I)

Vigenère analysis

Demo in CrypTool

The result of the Vigenère analysis can be manually reworked (changing the key length):

1. Encrypt starting example with **TEST**
   - *„Crypt/Decrypt“ \ „Symmetric (classic)“ \ „Vigenère…“*
   - Enter TEST *„Encrypt“*

   Analysis of the encryption
   - *„Analysis“ \ „Symmetric Encryption (classic)“ \ „Ciphertext only“ \ „Vigenère“*
   - Derived key length 4, Derived key TEST ✓

2. Encrypt starting example with **TESTETE**
   - *„Crypt/Decrypt“ \ „Symmetric (classic)“ \ „Vigenère…“*
   - Enter TESTETE *„Encrypt“*

   Analysis of the encryption
   - *„Analysis“ \ „Symmetric Encryption (classic)“ \ „Ciphertext only“ \ „Vigenère“*
   - Derived key length 5 – not correct ✗
   - Key length manually set to 7
   - Derived key TESTETE ✓

Demonstration of Interactivity (II)

Automated factorisation

Demo in CrypTool

Factorisation of a compound number with factorisation algorithms

- „Indiv. Procedures“ \ „RSA Cryptosystem“ \ „Factorisation of a Number“
- Some methods are executed in parallel (multi threaded)
- Methods have specific advantages and disadvantages (e.g. some methods can only determine small factors)

Factorisation example 1:

316775895367314538931177095642205088158145887517

48-digit decimal number

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Factorisation example 2:

2^250 - 1

75-digit decimal number

=

3 * 11 * 31 * 251 * 601 * 1801 * 4051 * 229668251 * 269089806001 *
4710883168879506001 * 5519485418336288303251

Concepts for a user-friendly Interface

1. Context sensitive help (F1)

- F1 on a selected menu entry shows information about the algorithm/method.
- F1 in a dialog box explains the use of the dialog.
- These assistances and the contents of the superordinate menus are cross linked in the online help.

2. Paste of keys in key-input dialog

- CTRL-V can be used to paste contents from the clipboard.
- Used keys can be taken out of cipher text windows via an icon in the icon bar. A corresponding icon in the key-input dialog can be used to paste the key into the key field. A CrypTool-internal memory which is available for every method is used (helpful for „specific“ keys – e.g. homophone encryption).

Challenges for Developers (Examples)

1. Many functions running parallel

- Factorisation runs with multi threaded algorithms

2. High performance

- Birthday paradox to locate hash collisions or for brute force analysis

3. Consider memory limits

- Floyd-algorithm (mappings to locate hash collisions) or Quadratic Sieve

4. Time measurement and estimates

- Display of elapsed time while using brute force

5. Reusability / Integration

- Forms for prime number generation
- RSA cryptosystem (switches after successful attack from PubKey view to PrivKey user)

Content

CrypTool Examples

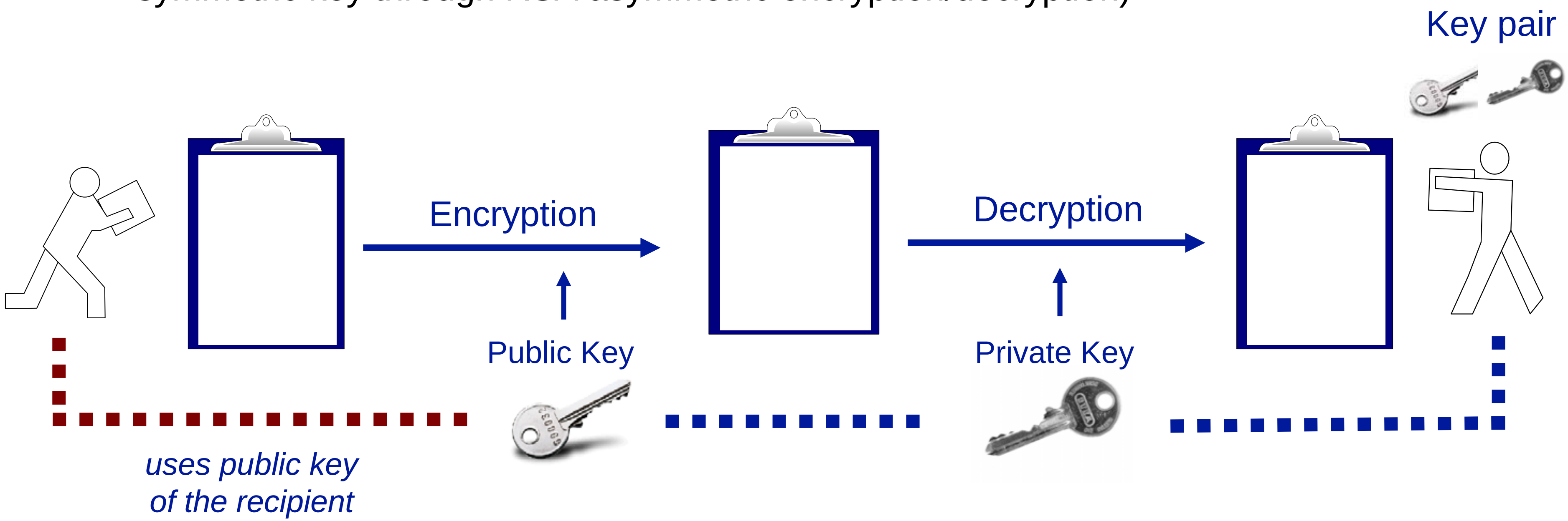
Overview of Examples

1. Encryption with RSA / Prime number test / Hybrid encryption and digital certificates
2. Digital signature visualised
3. Attack on RSA encryption (modul N too short)
4. Analysis of encryption in PSION 5
5. Weak DES keys
6. Locating key material (“NSA-Key”)
7. Attack on digital signature through location of hash collision
8. Authentication in a client-server environment
9. Demonstration of a side channel attack (on hybrid encryption protocol)
10. Attack on RSA using lattice reduction
11. Random analysis with 3-D visualisation
12. Secret Sharing (Chinese Remainder Theorem (CRT) / Shamir)
13. Implementation of CRT in Astronomy
14. Visualisation of symmetric encryption methods using ANIMAL
15. Generation of a message authentication code (MAC)
16. Hash Demo

Examples (I)

Encryption with RSA (in reality mostly hybrid encryption)

- **Basis** for e.g. SSL protocol (access to protected web sites)
- **Asymmetric encryption using RSA**
  - Every user has a key pair – one public and one private key
  - Sender encrypts with public key of the recipient
  - Recipient decrypts with his private key
- Implemented usually in a combination with symmetric methods (transfer of the symmetric key through RSA asymmetric encryption/decryption)



Examples (I)

Encryption using RSA – Mathematical background / algorithm

- Public key: (n, e)
- Private key: (d)

where:

p, q large, randomly chosen prime numbers with $n = p*q$;

d is calculated under the constraints $\gcd[\varphi(n),e] = 1$; $e*d \equiv 1 \mod \varphi(n)$.

Encryption and decryption operation: $(m^e)^d \equiv m \mod n$

- n is the module, which length in bits is referred to as RSA key length.
- gcd = greatest common divisor.
- $\varphi(n)$ is the Euler phi function.

Procedure:

- Transformation of message in binary representation
- Encrypt message $m = m_1,...,m_k$ block wise, with for all m_j:
  $0 \leq m_j < n$; maximum block size r, so that: $2^r \leq n$

Examples (I)

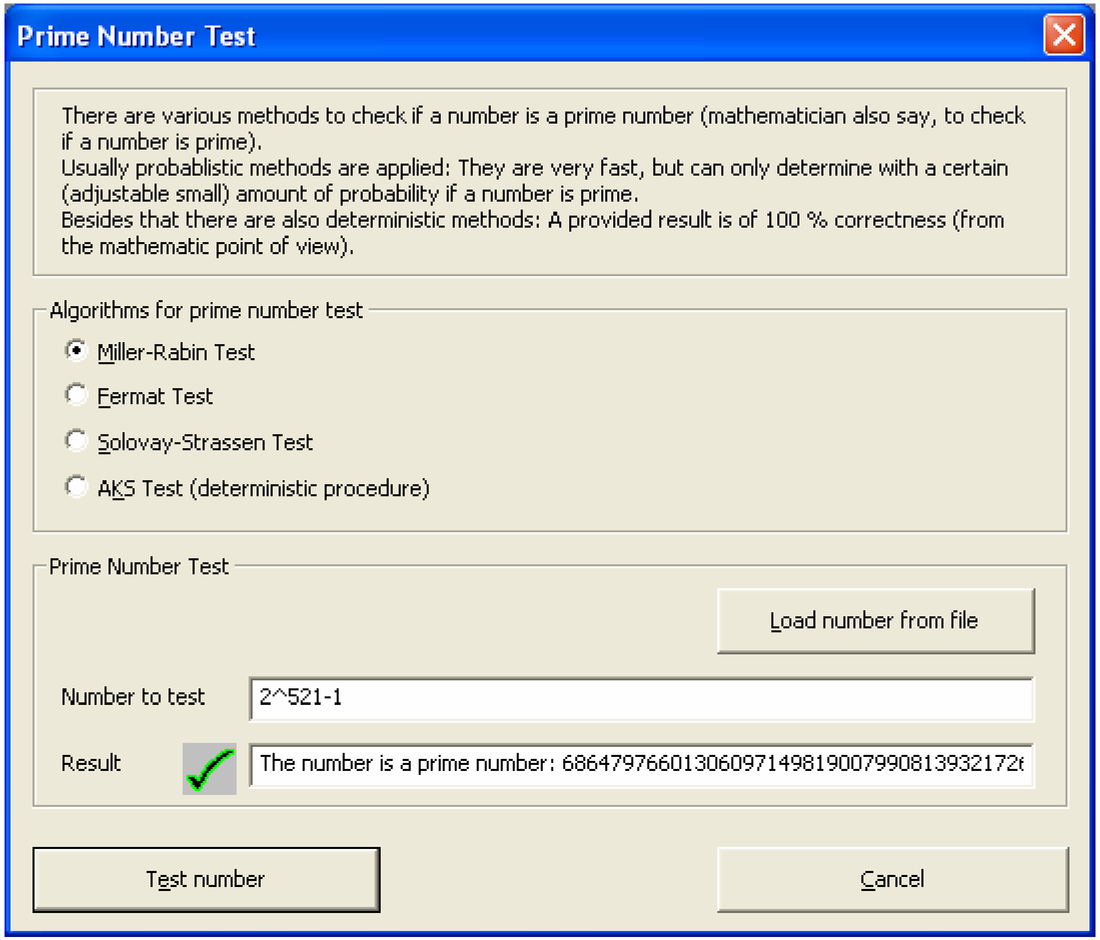
Prime number tests – For RSA huge primes are needed.

- Fast probabilistic tests
- Deterministic tests

The prime number test methods can test much faster whether a big number is prime,

than the known factorization methods can divide a number of a similar size in its prime factors.

For the AKS method the GMP library (**G**NU **M**ultiple **P**recision Arithmetic Library) is integrated into CrypTool.



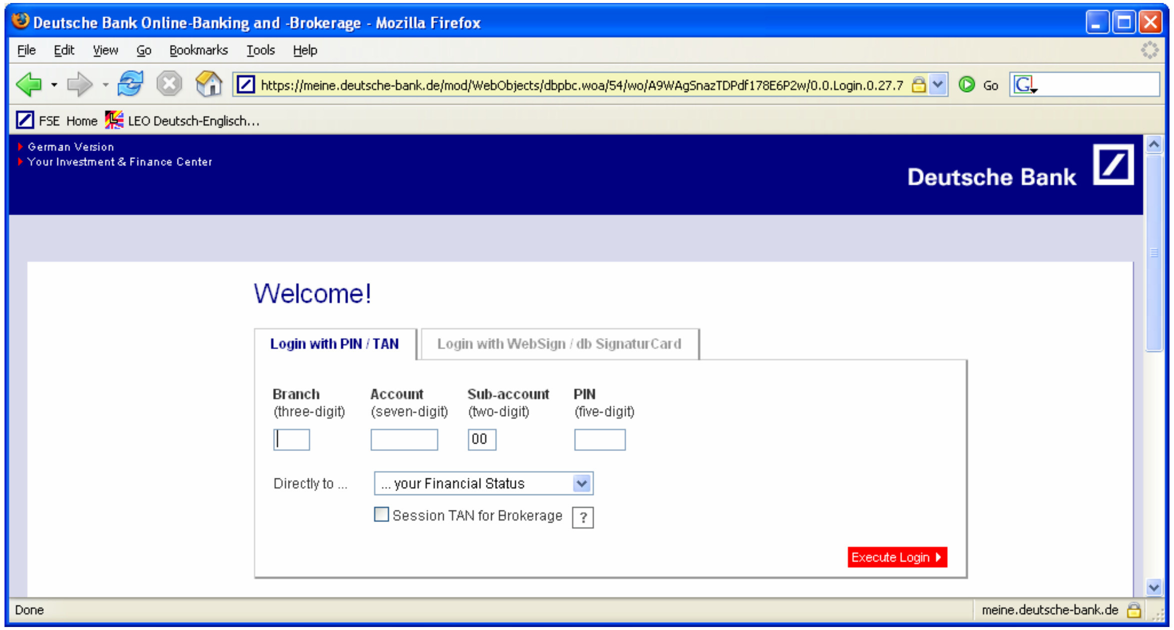
Example (I)

Hybrid encryption and digital certificates

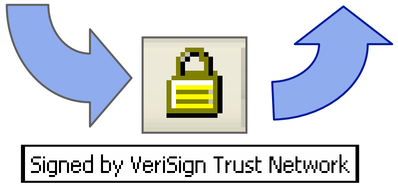
- **Hybrid encryption** – Combination of asymmetric and symmetric encryption
  1. Generation of a random symmetric key (Session Key)
  2. Session key is transferred – protected by asymmetric key
  3. Message is transferred – protected by session Key
- **Problem**: Man-in-the-middle attacks – does the public key of the recipient really belong to the recipient?
- **Solution: Digital certificates** – A central instance (e.g. VeriSign, Deutsche Bank PKI), that is being trusted by all users, ensures the authenticity of the certificate and the contained public key (similar to a passport issued by the state).
- **Hybrid encryption based on digital certificates** is the foundation for all secured electronic communication (e.g. SSL):
  - Internet Shopping and Online Banking
  - Secure eMail

Example (I)

Secured online connection using SSL and certificates

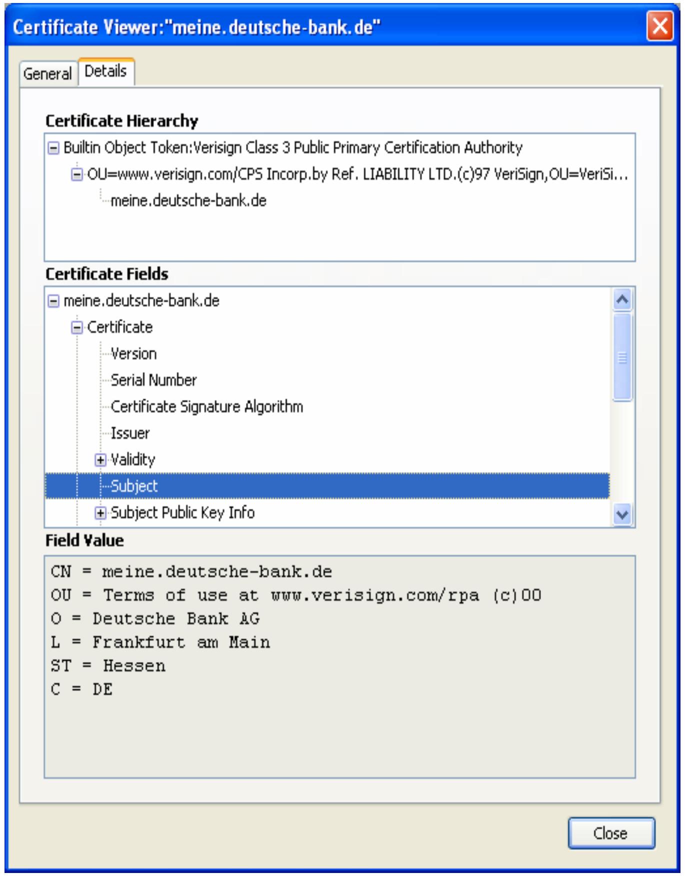


This means, that the connection is authenticated (at least at one side) and that the transferred data are strongly encrypted.



Example (I)

Attributes or fields of a certificate



General attributes / fields

- Issuer (e.g. VeriSign)
- Requestor
- Validity period
- Serial number
- Certificate type / Version (X.509v3)
- Signature algorithm
- Public key (and method)

Public Key

Examples (II)

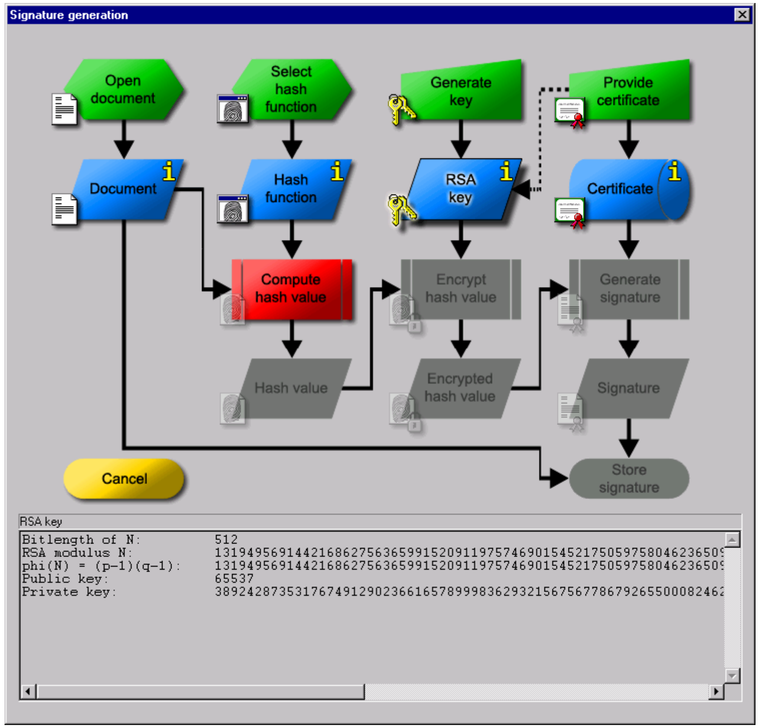
Digital signature visualised

Digital signature

- Increasingly important
  - equivalence with manual signature (digital signature law)
  - increasingly used by industry, government and consumers
- Few people know how it works

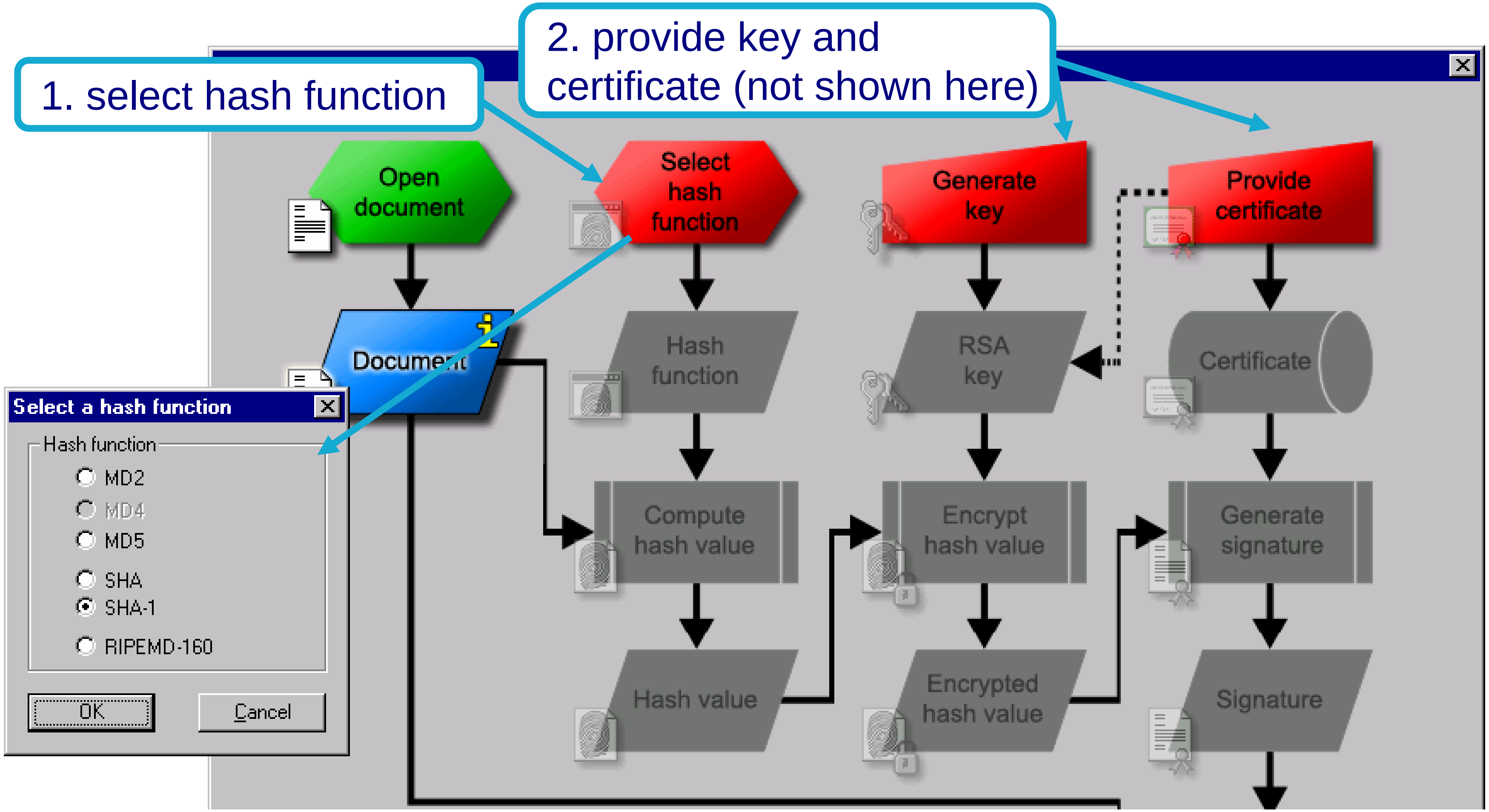
Visualisation in CrypTool

- See menu “Digital Signatures/PKI” \ “*Signature Demonstration (Signature Generation)*“
- Interactive data flow diagram
- Similar to the visualisation of hybrid encryption



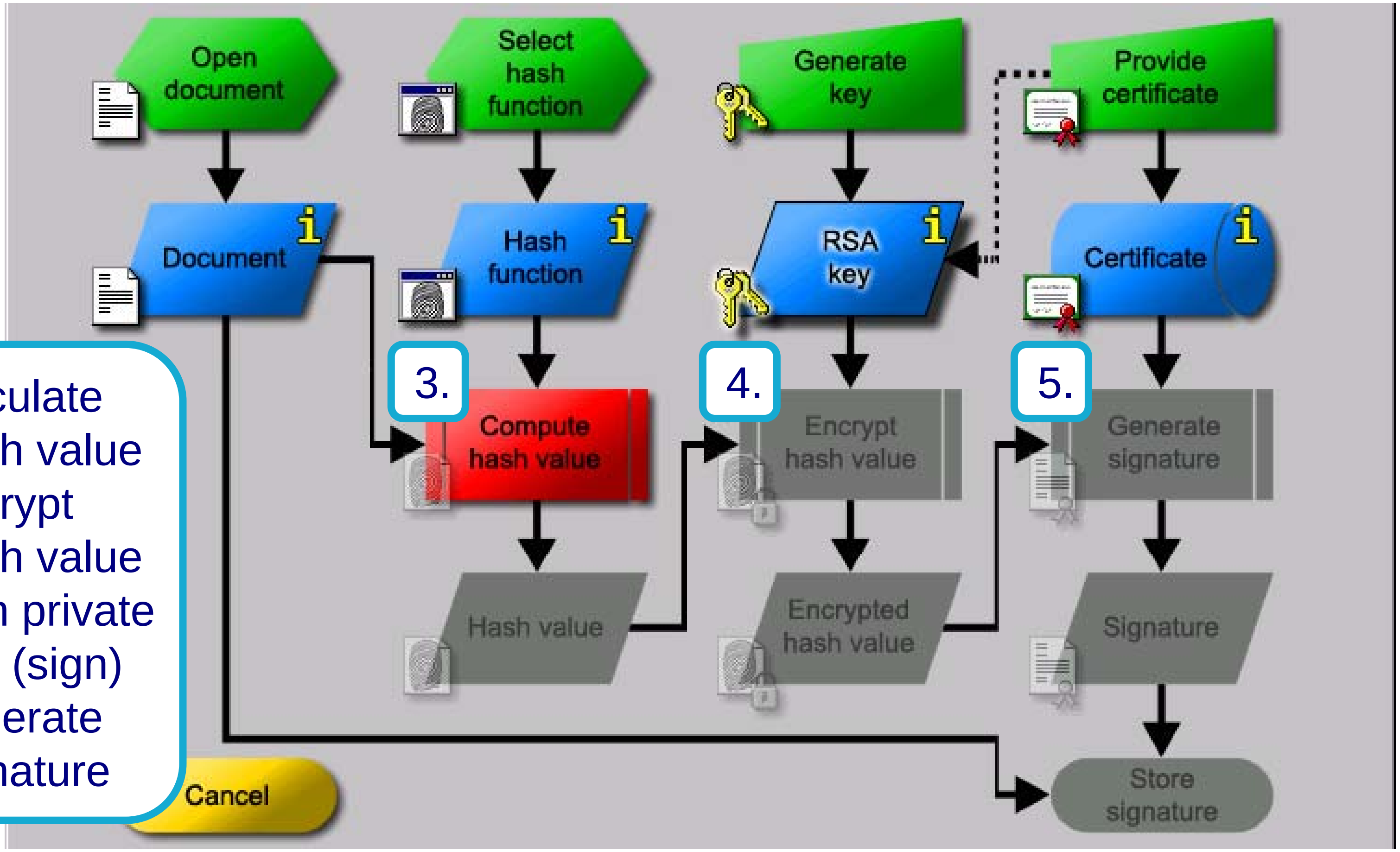
Examples (II)

Digital signature visualised: a) Preparation



Examples (II)

Digital signature visualised: b) Cryptography



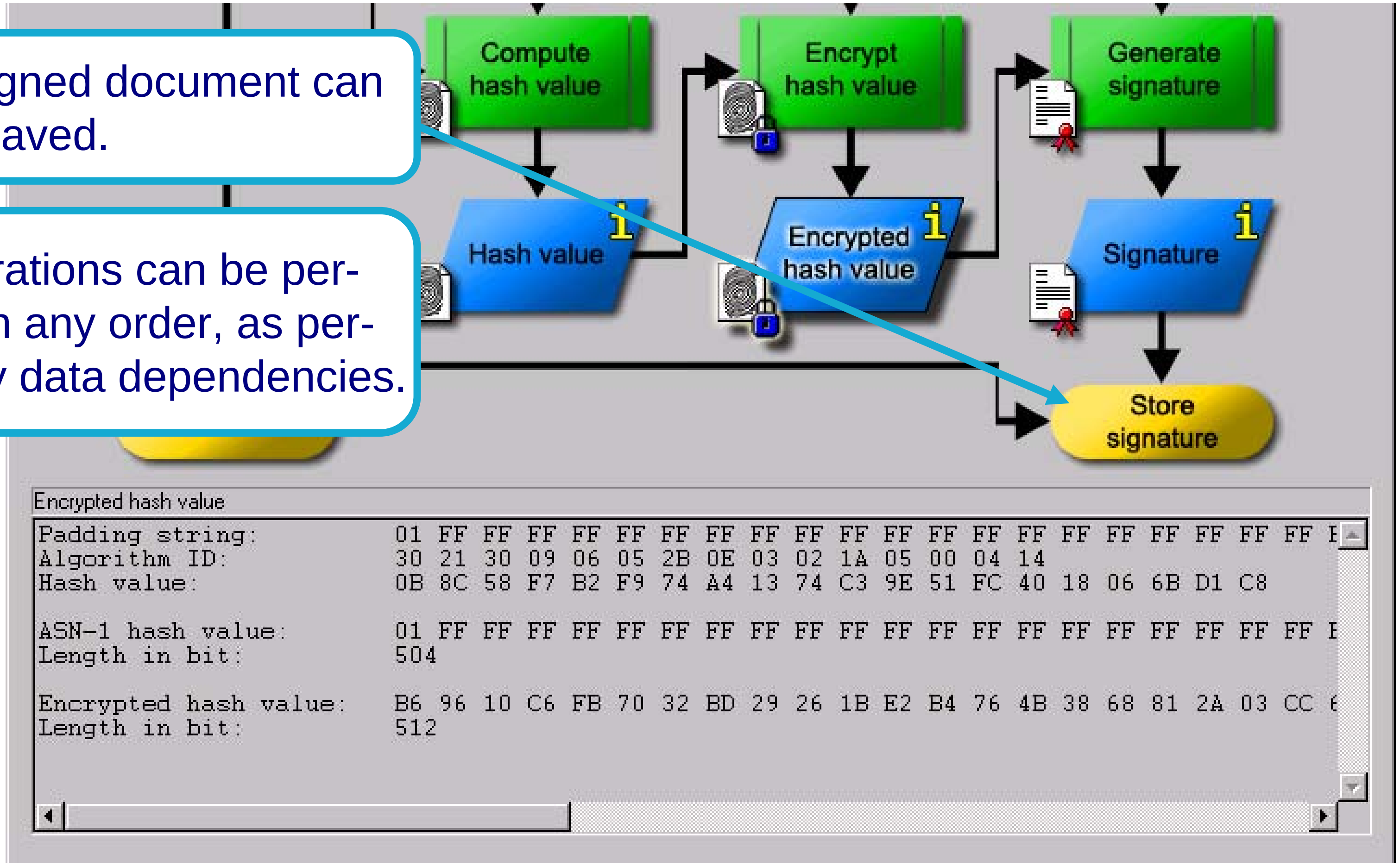
3. calculate hash value
4. encrypt hash value with private key (sign)
5. generate signature

Examples (II)

Digital signature visualised: c) Result

6. The signed document can now be saved.

The operations can be performed in any order, as permitted by data dependencies.



Examples (III)

Attack on RSA encryption with short RSA modulus

Example from *Song Y. Yan*, Number Theory for Computing, Springer, 2000

- public key
  - RSA modulus N = 63978486879527143858831415041 (95 bit, 29 decimal digits)
  - public exponent e = 17579
- cipher text (block length = 14):
  - C_1 = 45411667895024938209259253423,
    C_2 = 16597091621432020076311552201,
    C_3 = 46468979279750354732637631044,
    C_4 = 32870167545903741339819671379
- the text shall be deciphered!

The ciphertext is not necessary for the actual cryptanalysis (locating the private key) !

Solution using CrypTool (more detailed in online help examples section):

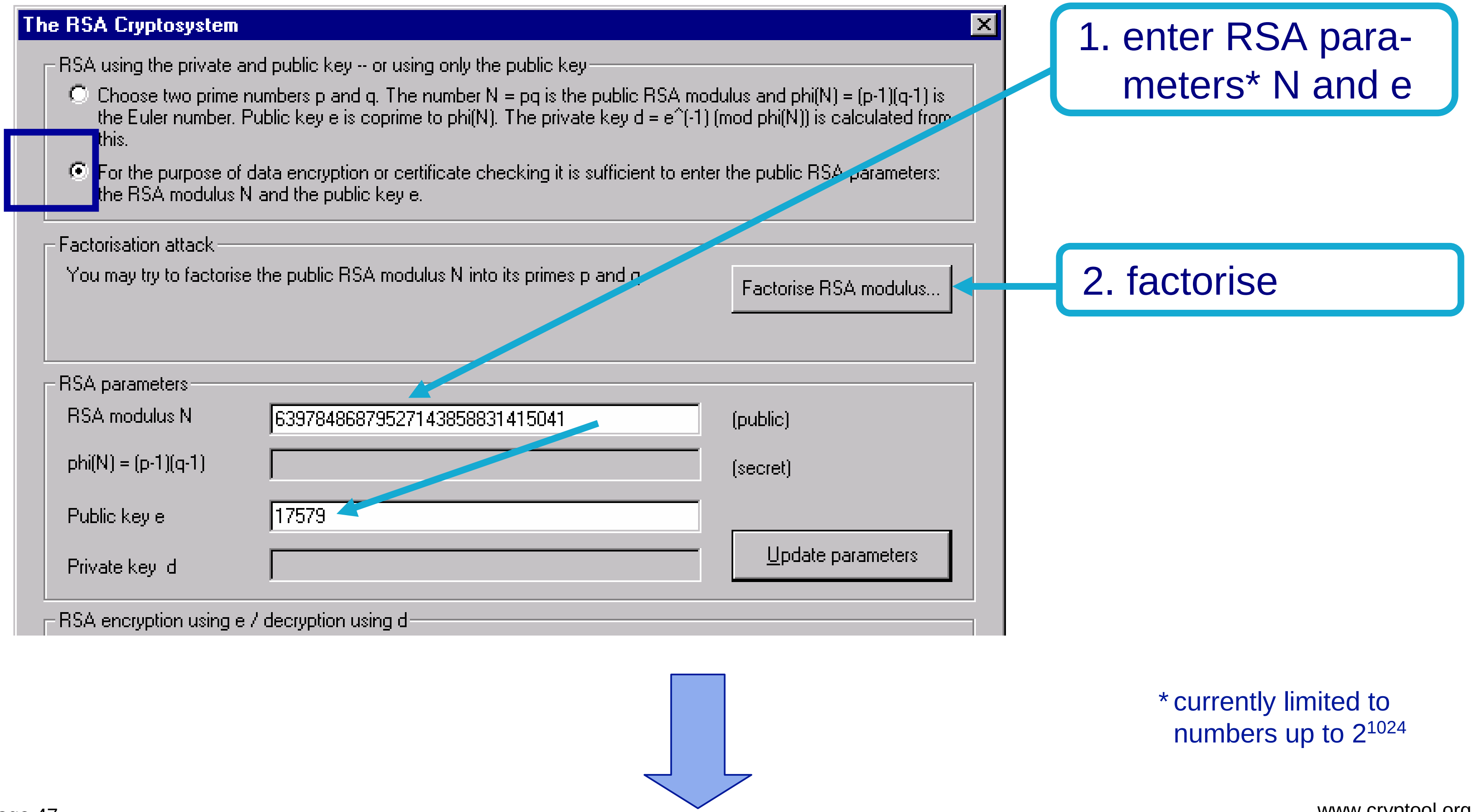
- enter public parameters into "RSA cryptosystem" (menu Indiv. Procedures)
- button "factorise the RSA modulus" yields prime factors pq = N
- based on that information private exponent $d=e^{-1} \bmod (p-1)(q-1)$ is determined
- decrypt the cipher text with d: $M_i = C_i^d \bmod N$

The attack with CrypTool is workable for RSA moduli up to 250 bit

Then you could digitally sign for someone else !

Examples (III)

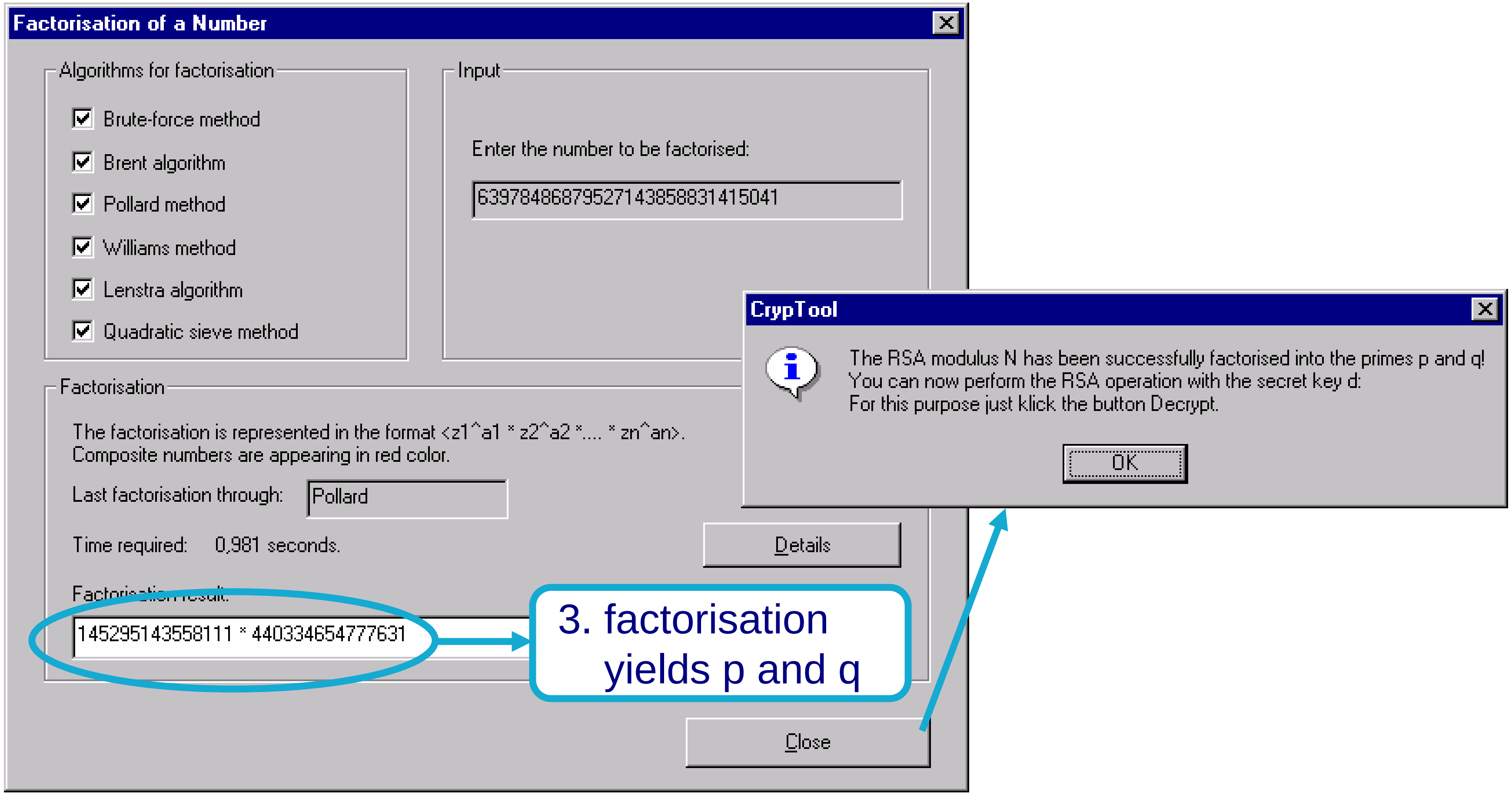
Short RSA modulus: enter public RSA parameters



* currently limited to numbers up to 2^{1024}

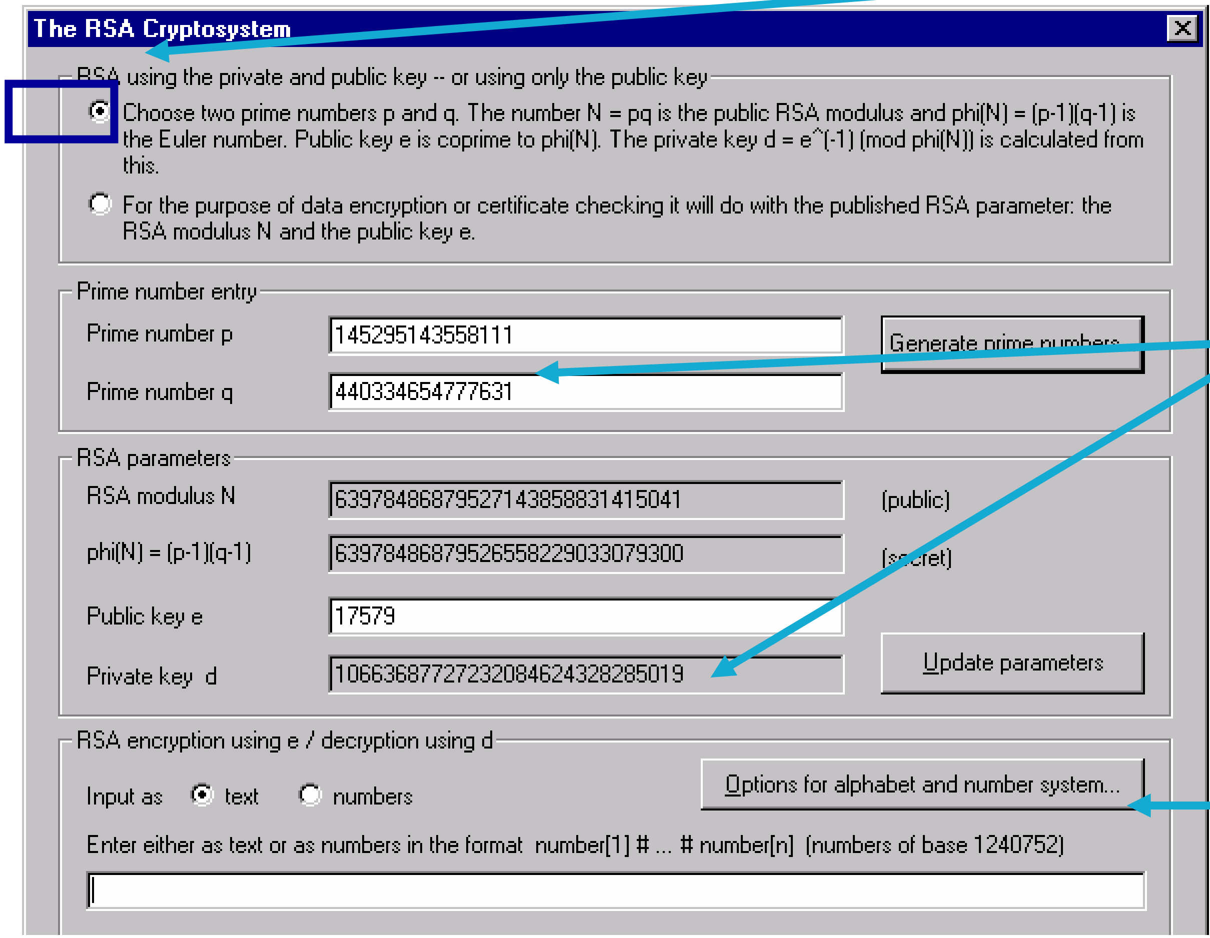
Examples (III)

Short RSA modulus: factorise RSA modulus



Examples (III)

Short RSA modulus: determine private key d



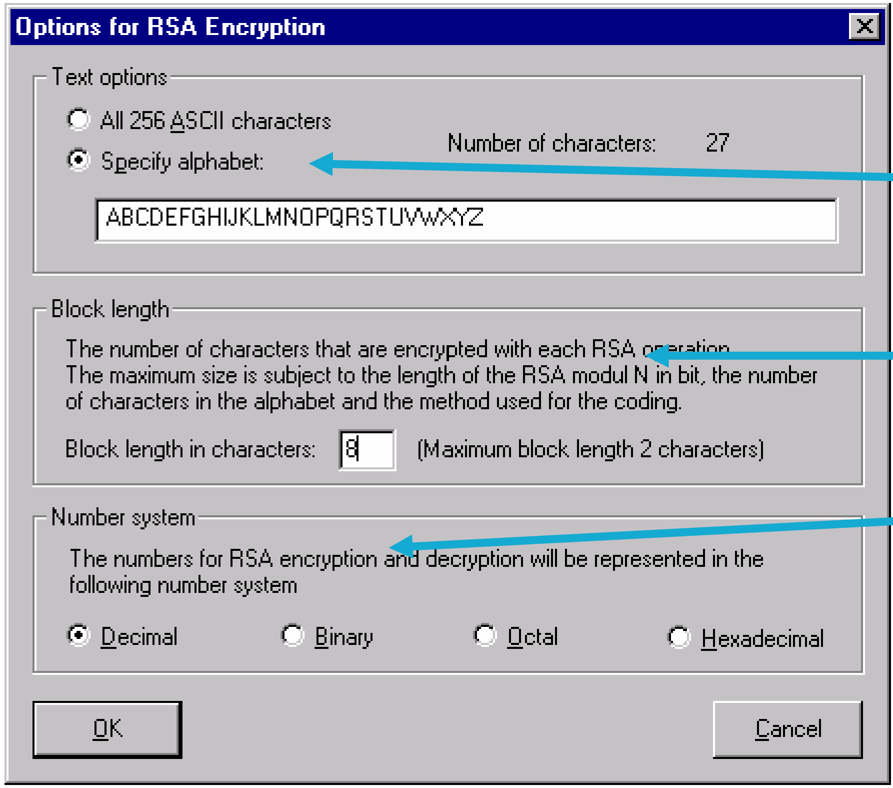
Change the view to the owner of the secret key.

4. p and q have been entered automatically and secret key d has been calculated

5. adjust options

Examples (III)

Short RSA modulus: adjust options



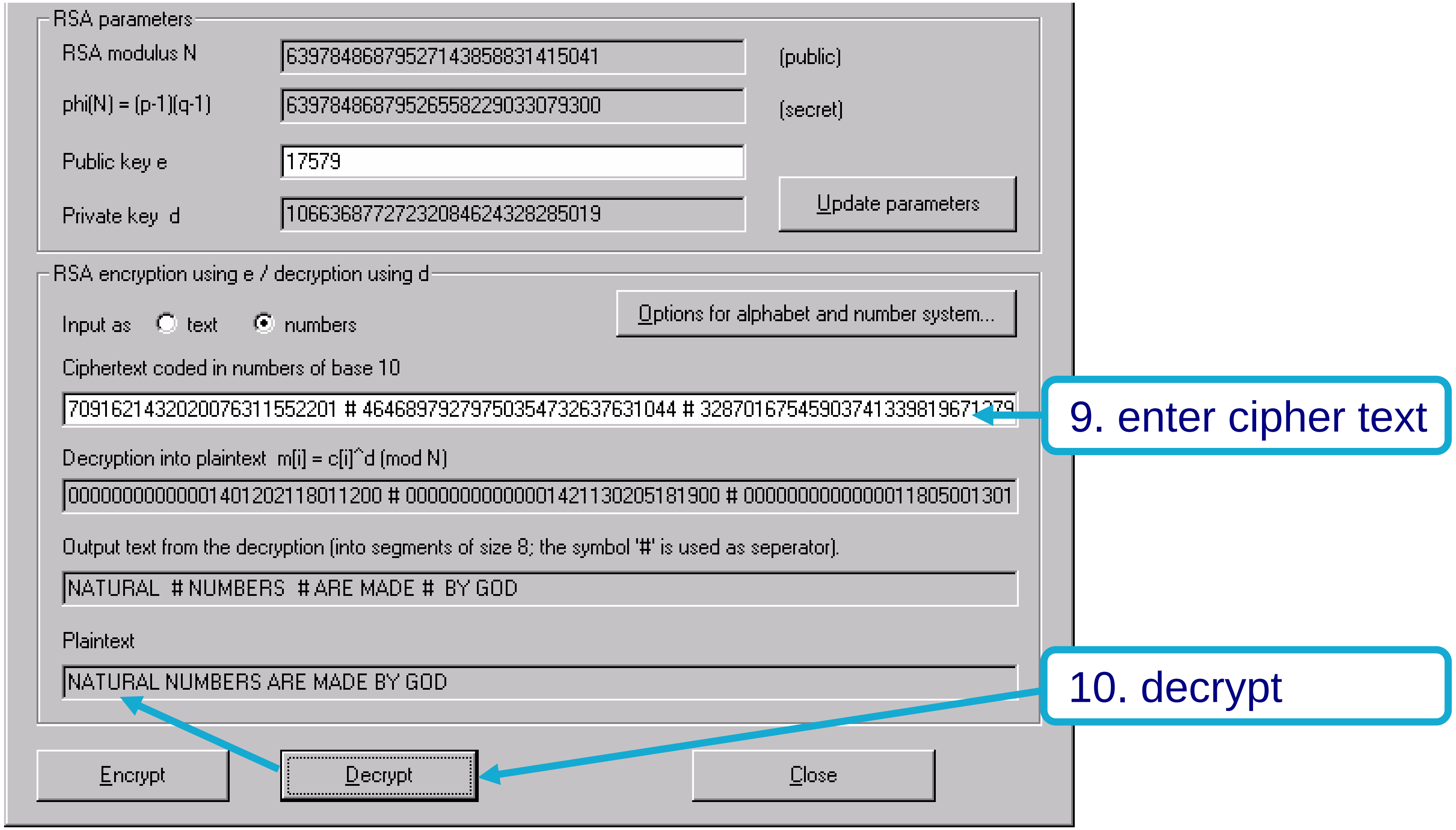
6. select alphabet

7. select coding method

8. select block length

Examples (III)

Short RSA modulus: decrypt cipher text



Examples (IV)

Analysis of encryption used in the PSION 5

Practical application of cryptanalysis:

Attack on the encryption option in the PSION 5 PDA word processing application

Starting point: an encrypted file on the PSION

Requirements

- encrypted English or German text
- depending on method and key length, 100 bytes up to several kB of text

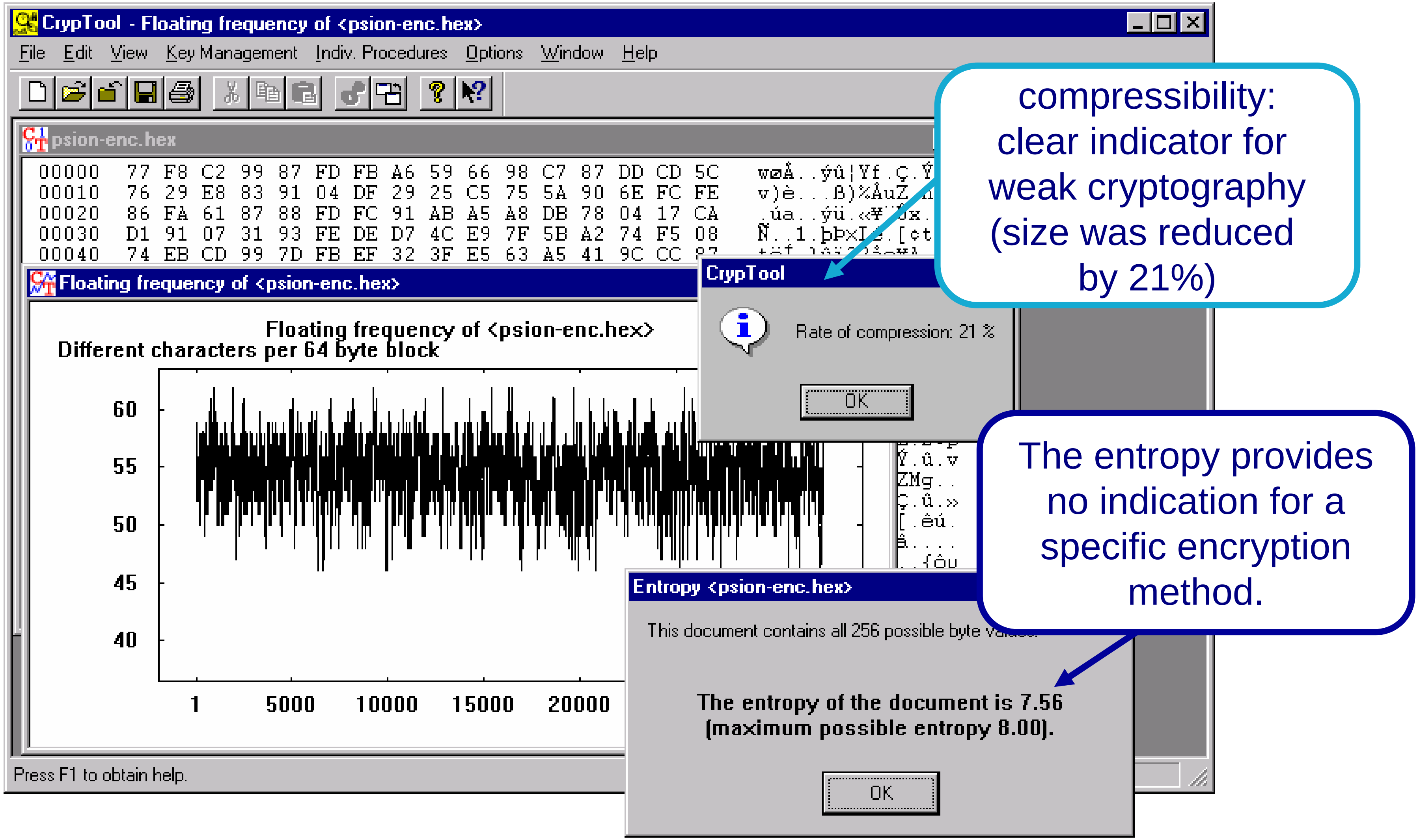
Procedure

- pre-analysis
  - entropy
  - floating entropy
  - compression test
- auto-correlation
- try out automatic analysis with classical methods

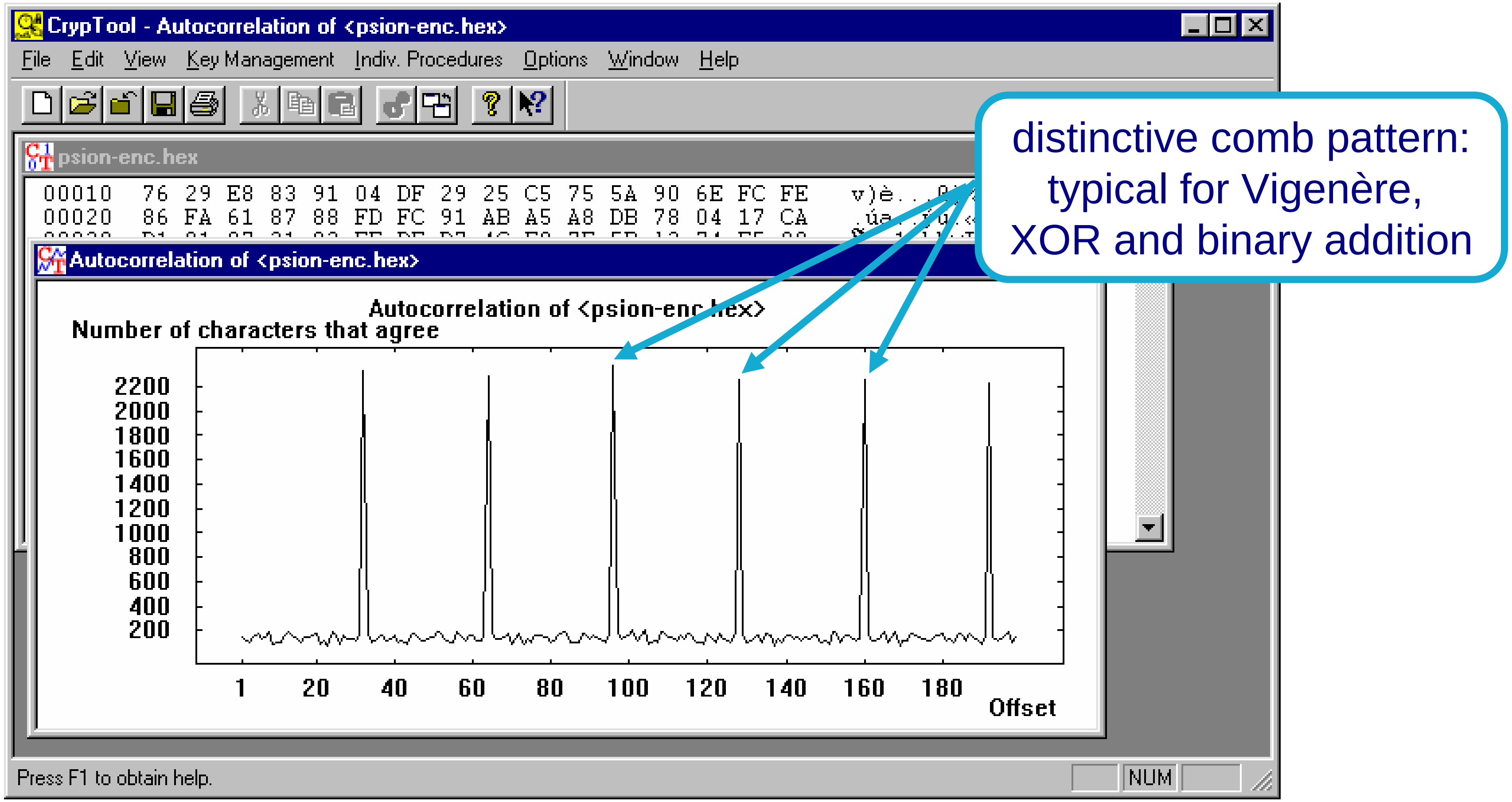
$\Rightarrow$ probably classical encryption algorithm

Examples (IV)

PSION 5 PDA – determine entropy, compression test



Examples (IV)
PSION 5 PDA – determine auto-correlation



* The encrypted file is available with CrypTool (see CrypTool\examples\psion-enc.hex)

Examples (IV)

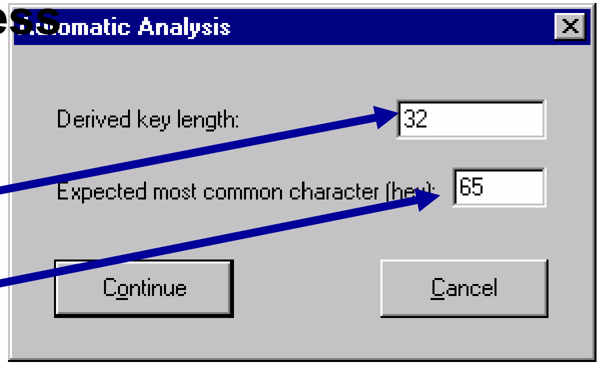
PSION 5 PDA – automatic analysis

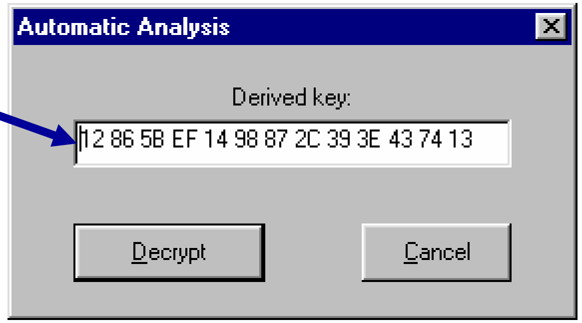
Automatic analysis using Vigenère: no success

Automatic analysis using XOR: no success

Automatic analysis using binary addition:

- CrypTool calculates the key length using auto-correlation: 32 bytes
- The user can choose which character is expected to occur most frequently: “e” = 0x65 (ASCII code)
- Analysis calculates the most likely key (based on the assumptions about distribution)
- Results: good, but not perfect



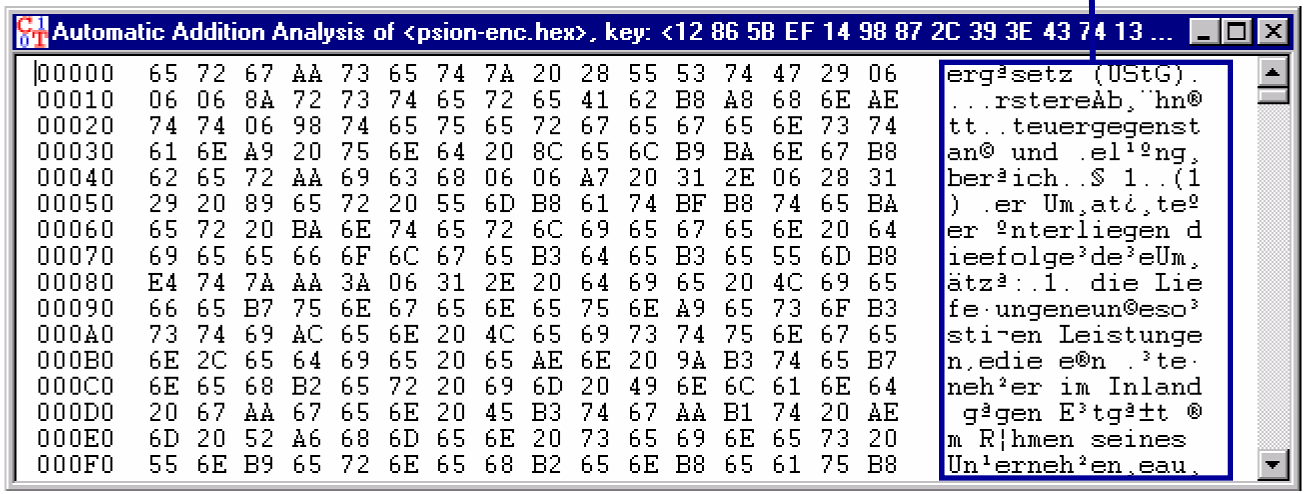


Examples (IV)

PSION 5 PDA – results of automatic analysis

Results of automatic analysis with assumption “binary addition”:

- results good, but not perfect: 24 out of 32 key bytes correct.
- the key length 32 was correctly determined.



- the password entered was not 32 bytes long.
  ⇒ PSION Word derives the actual key from the password.
- manual post-processing produces the encrypted text
  (not shown)

Examples (IV)

PSION 5 PDA – determining the remaining key bytes

Copy key to clipboard during automatic analysis

In automatic analysis hex dump,

- determine incorrect byte positions, e.g. 0xAA at position 3
- guess and write down corresponding correct bytes: „e“ = 0x65

In encrypted initial file hex dump,

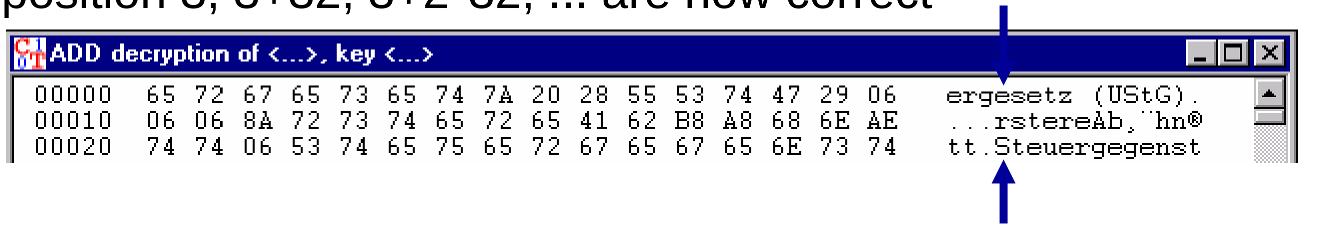
- determine initial bytes from the calculated byte positions: 0x99
- calculate correct key bytes with CALC.EXE: 0x99 - 0x65 = 0x34

Correct key from the clipboard

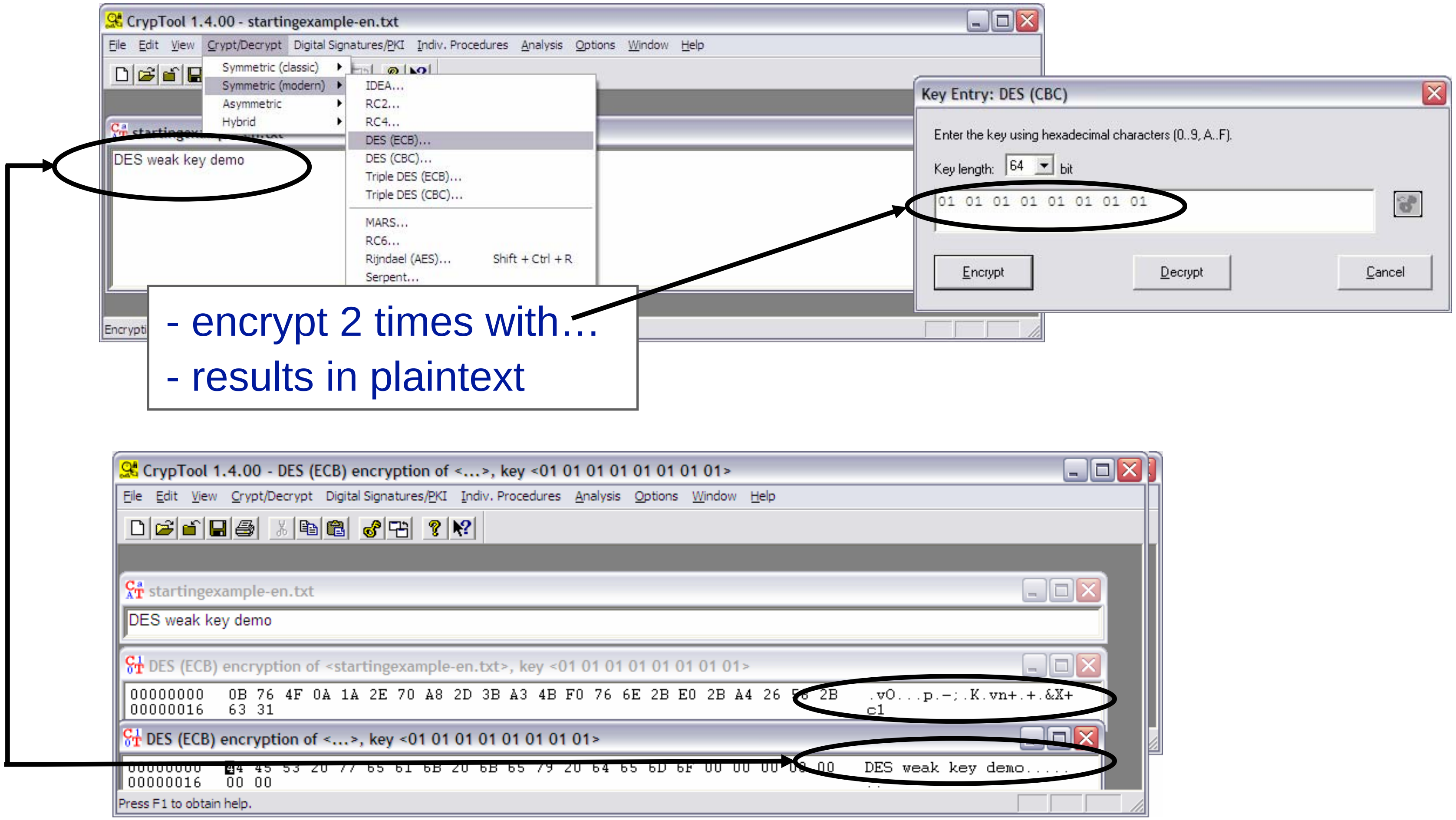
- 12865B341498872C393E43741396A45670235E111E907AB7C0841...

Decrypt encrypted initial document using binary addition

- bytes at position 3, 3+32, 3+2*32, ... are now correct



Examples (V)
Weak DES key



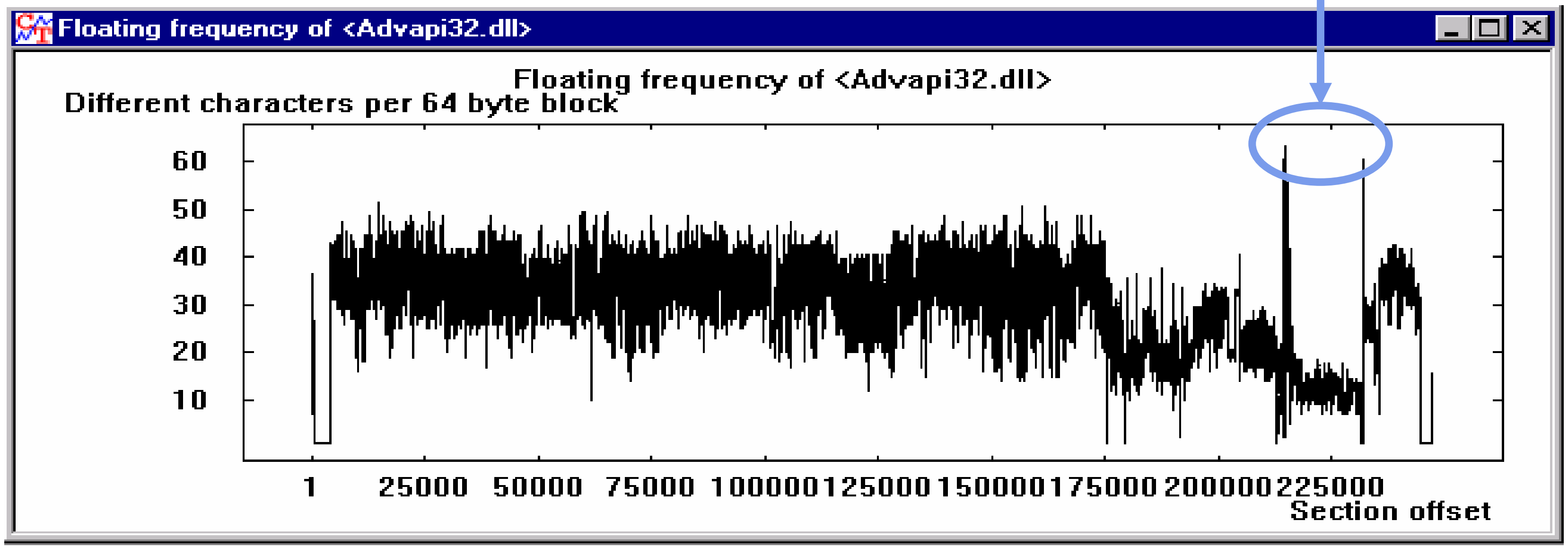
Examples (VI)

Locate key material

The function “Floating frequency” is suitable for locating key material and encrypted areas in files.

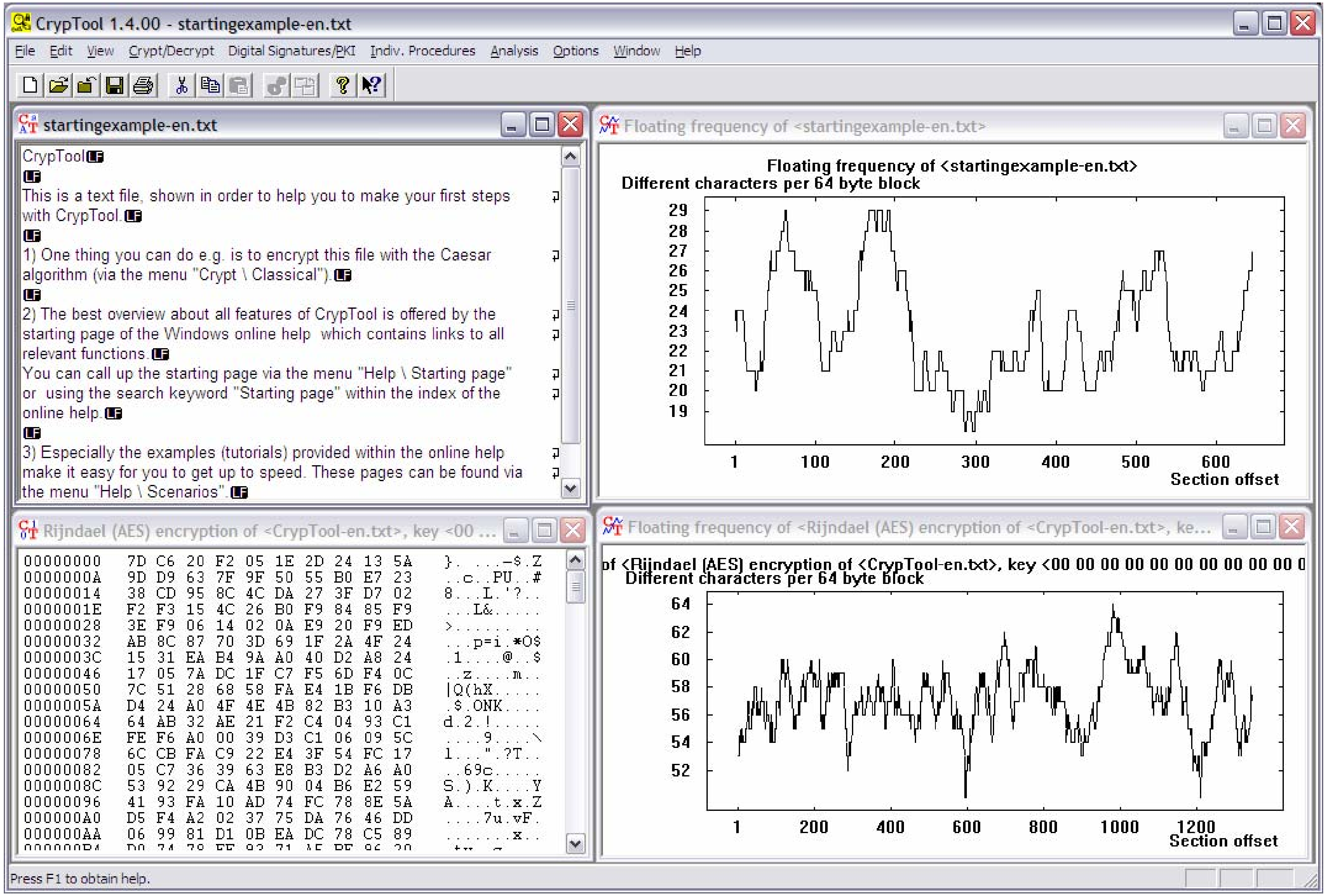
Background:

- key data is “more random” than text or program code
- can be recognised as peaks in the “floating frequency”
- example: the “NSAKEY” in advapi32.dll



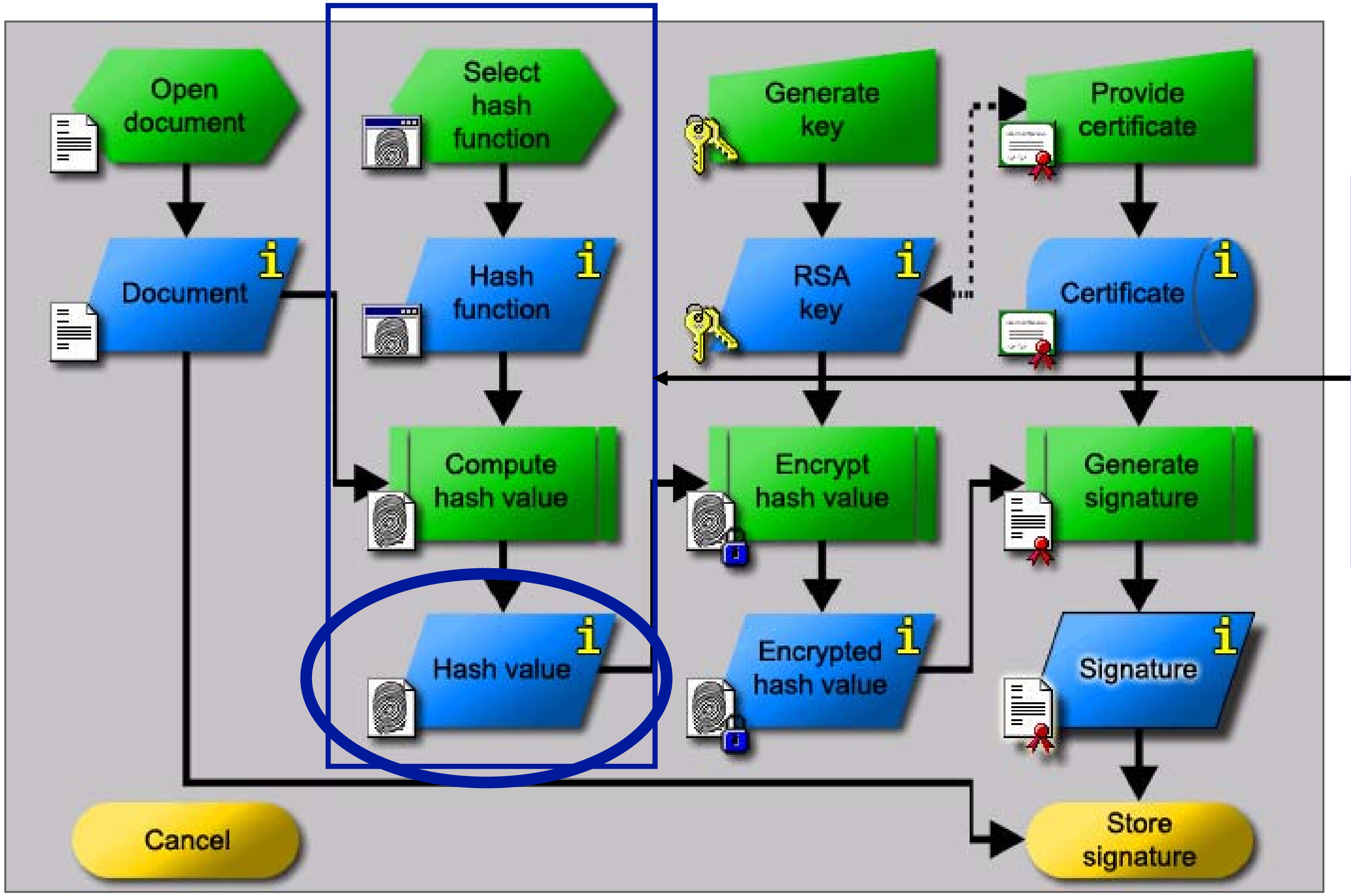
Examples (VI)

Comparison on floating frequency with other files



Examples (VII)

Attack on digital signature



Attack:

Find two messages with the same hash value !

Examples (VII)

Attack on digital signature – idea (I)

Attack on the digital signature of an ASCII text based on hash collision search

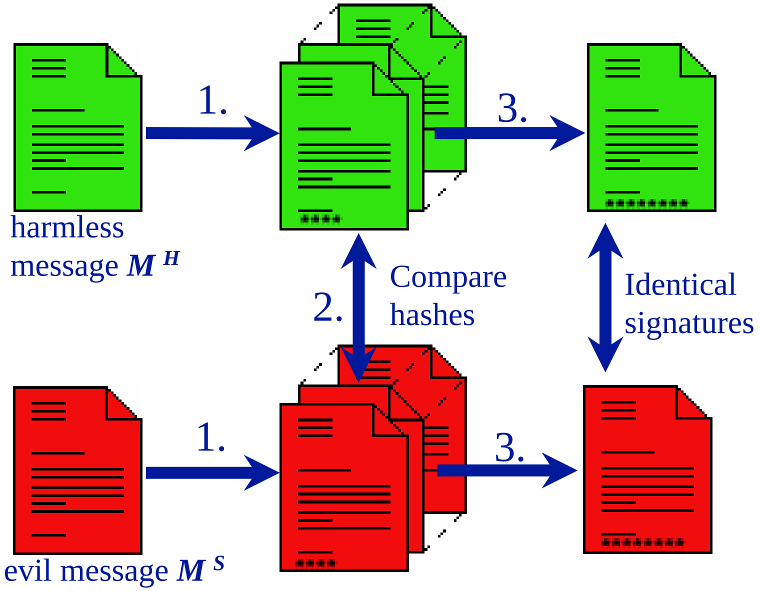
Idea:

- ASCII-Texts can be modified by changing/inserting ***non-printable*** characters, without changing the visible content
- modify two texts in parallel until a hash collision is found
- exploit the birthday paradox (birthday attack)
- generic attack applicable to all hash functions
- can be run in parallel on many machines (not implemented)
- implemented in CrypTool as part of his bachelor thesis “*Methods and tools for attacks on digital signatures*” (German), 2003.

Concepts: Mappings, modified Floyd algorithm (constant memory consumption) !

Examples (VII)

Attack on digital signature – idea (II)



1. **Modification:** starting from a message M create N different messages $M_1, ..., M_N$ with the same "content" as M.
2. **Search:** find modified messages M_i^H und M_j^S with the same hash value.
3. **Attack:** the signatures of those two documents M_i^H und M_j^S are the same.

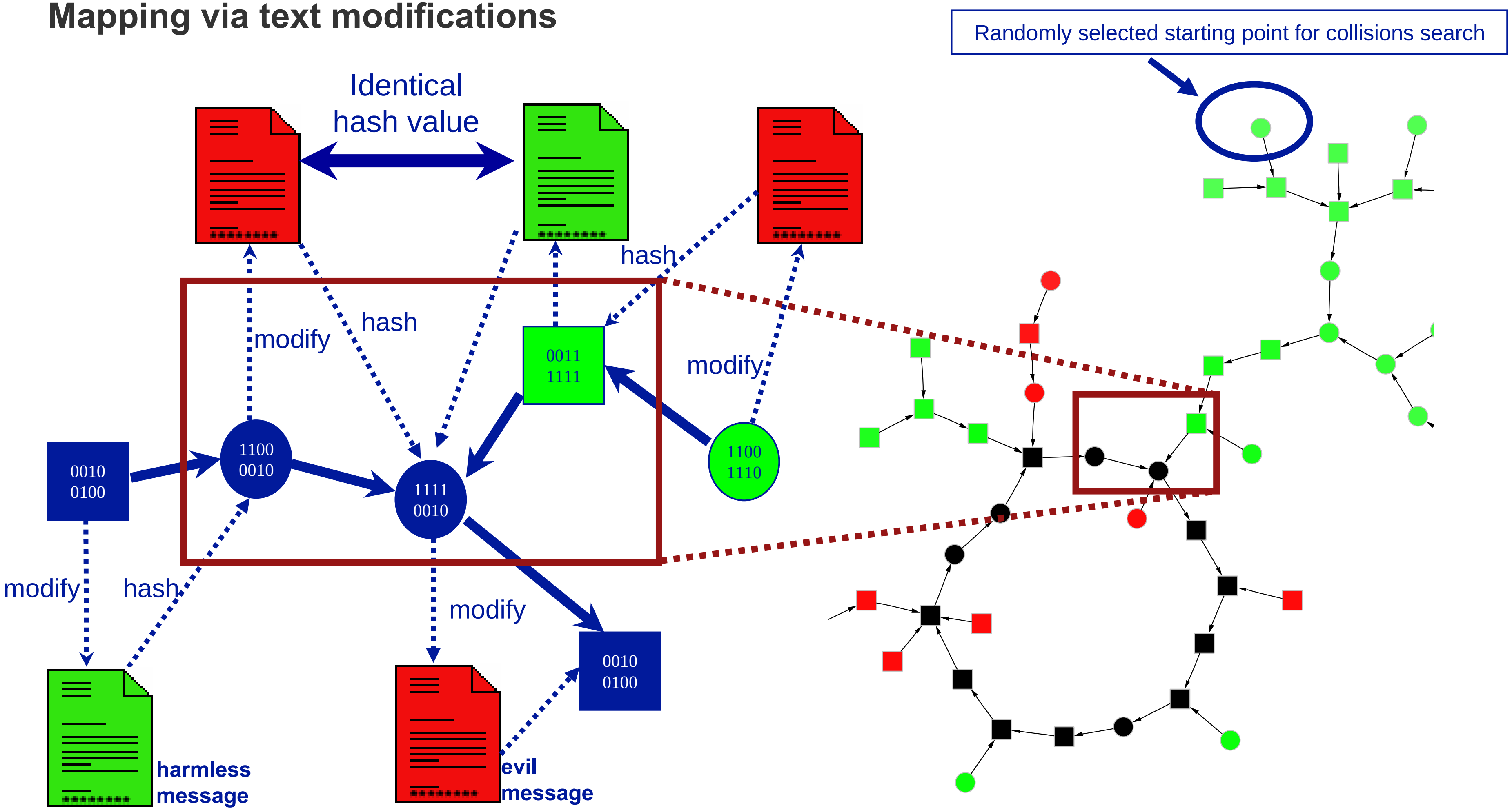
We know from the birthday paradox that for hash values of bit length n:

- search collision between M^H and $M_1^S, ..., M_N^S$: $N \approx 2^n$
- search collision between $M_1^H, ..., M_N^H$ and $M_1^S, ..., M_N^S$: $N \approx 2^{n/2}$

Estimated number of generated messages in order to find a hash collision.

Locate Hash Collisions

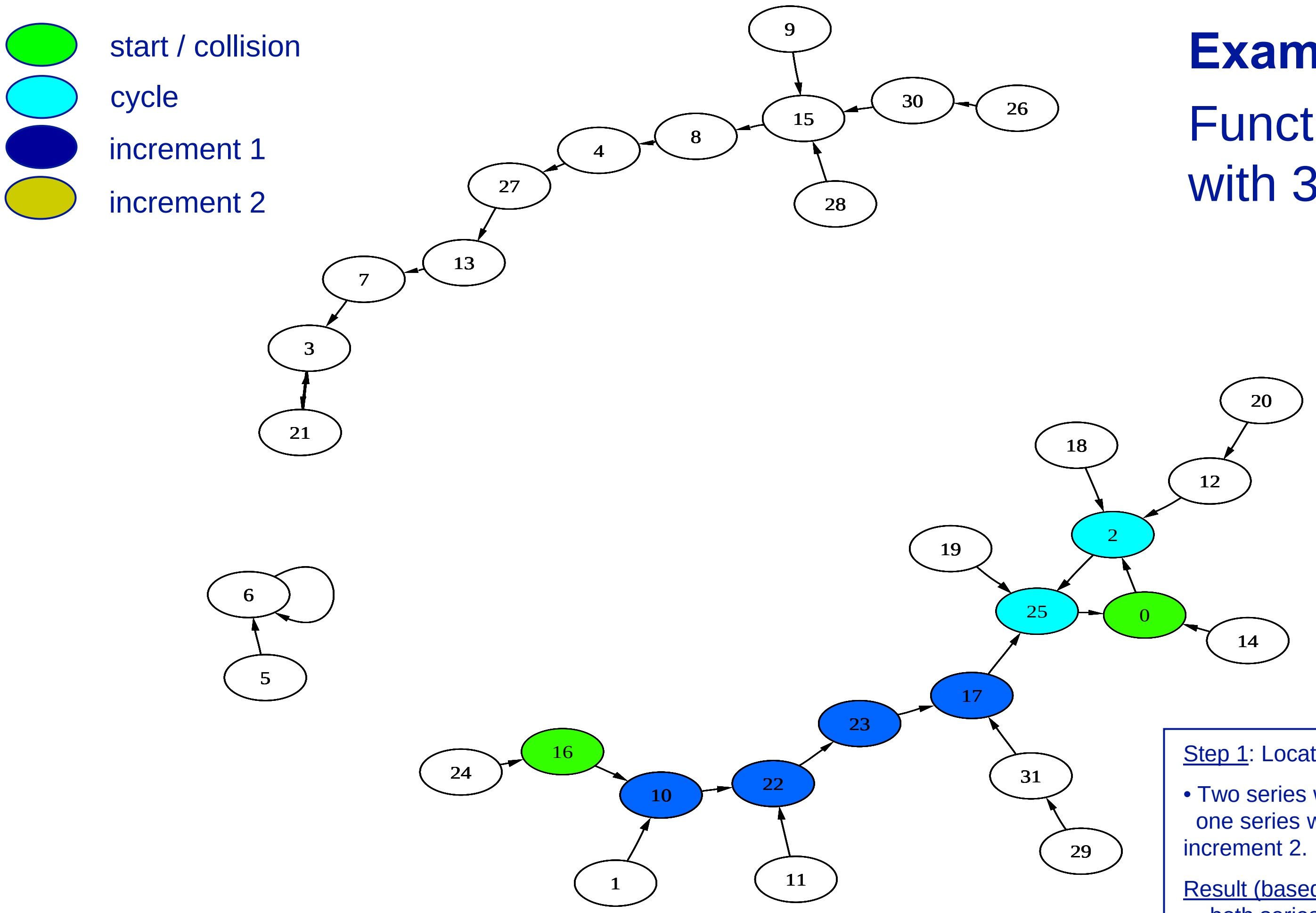
Mapping via text modifications



- green / red: path from a tree to the cycle – this can lead to a useful or useless collision.
- square / round: hash value has even / odd parity
- black: all nodes within the cycle

Locate Hash Collisions

Floyd-Algorithm: meet within the cycle



Example:

Function graph
with 32 nodes

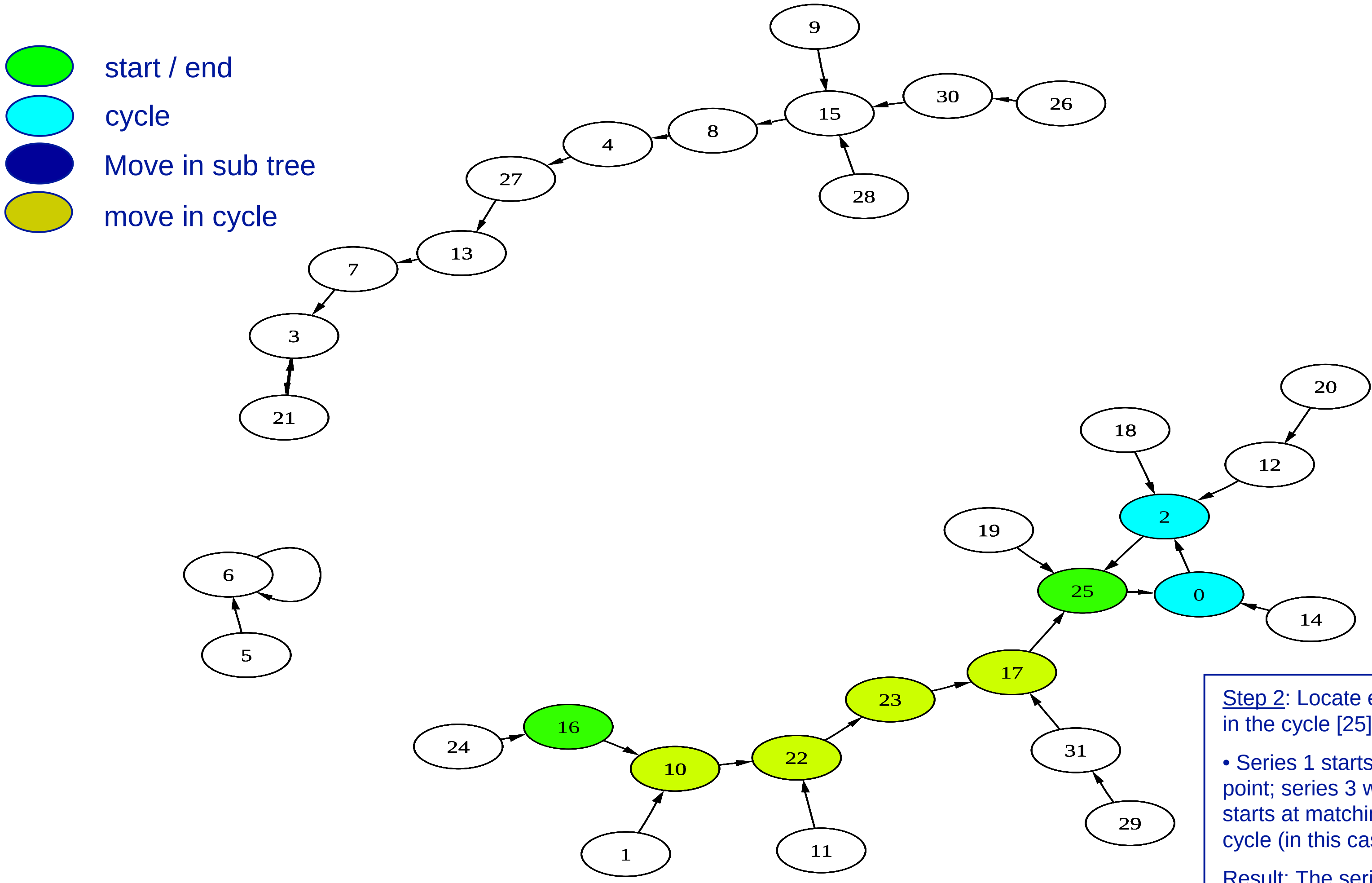
<u>Step 1</u>: Locate matching point within cycle:

• Two series with identical starting point [16]: one series with increment 1, the other with increment 2.

<u>Result (based on graph theory)</u>:
- both series always end up in a cycle.
- both series match in a node within the cycle (in this case 0).

Locate Hash Collisions

Step into cycle (Extension of Floyd): find entry point



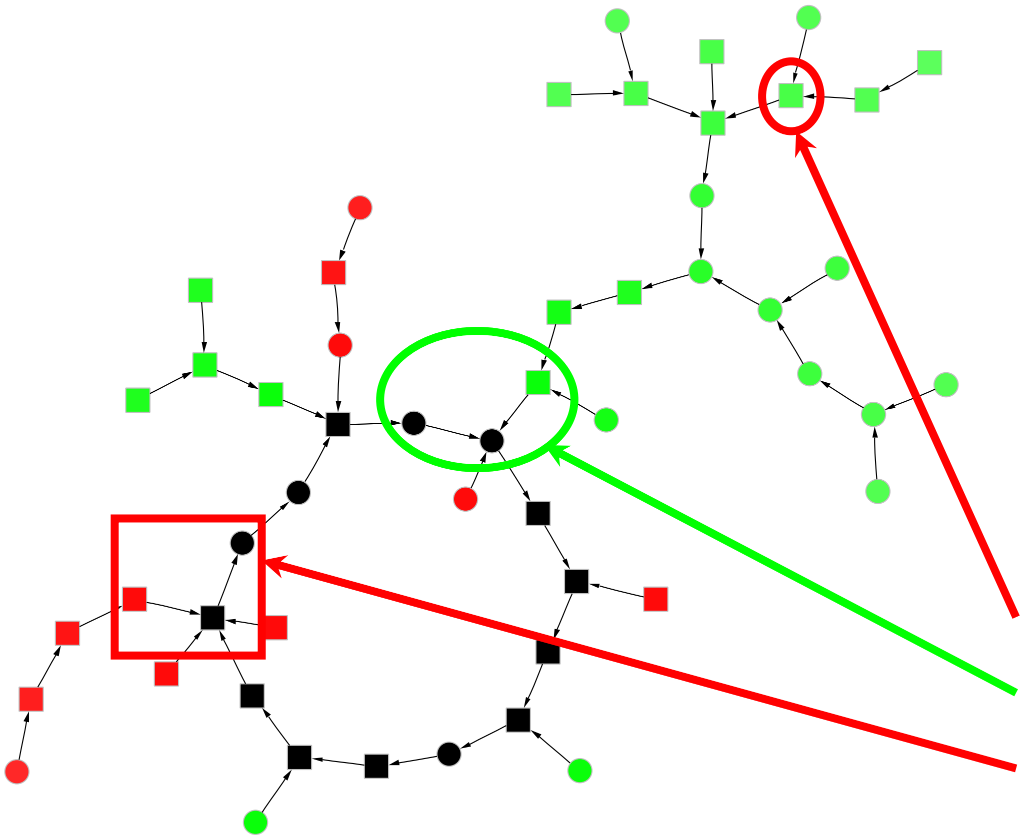
<u>Step 2</u>: Locate entry point of series 1 in the cycle [25]:

• Series 1 starts again from starting point; series 3 with an increment of 1 starts at matching point within the cycle (in this case 0).

<u>Result</u>: The series (1 and 3) match in cycle entry point of series 1 (in this case 25)

• The predecessors (in this case 17 and 2) result in a hash collision.

Birthday paradox attack on digital signature



Real examination of Floyd algorithm

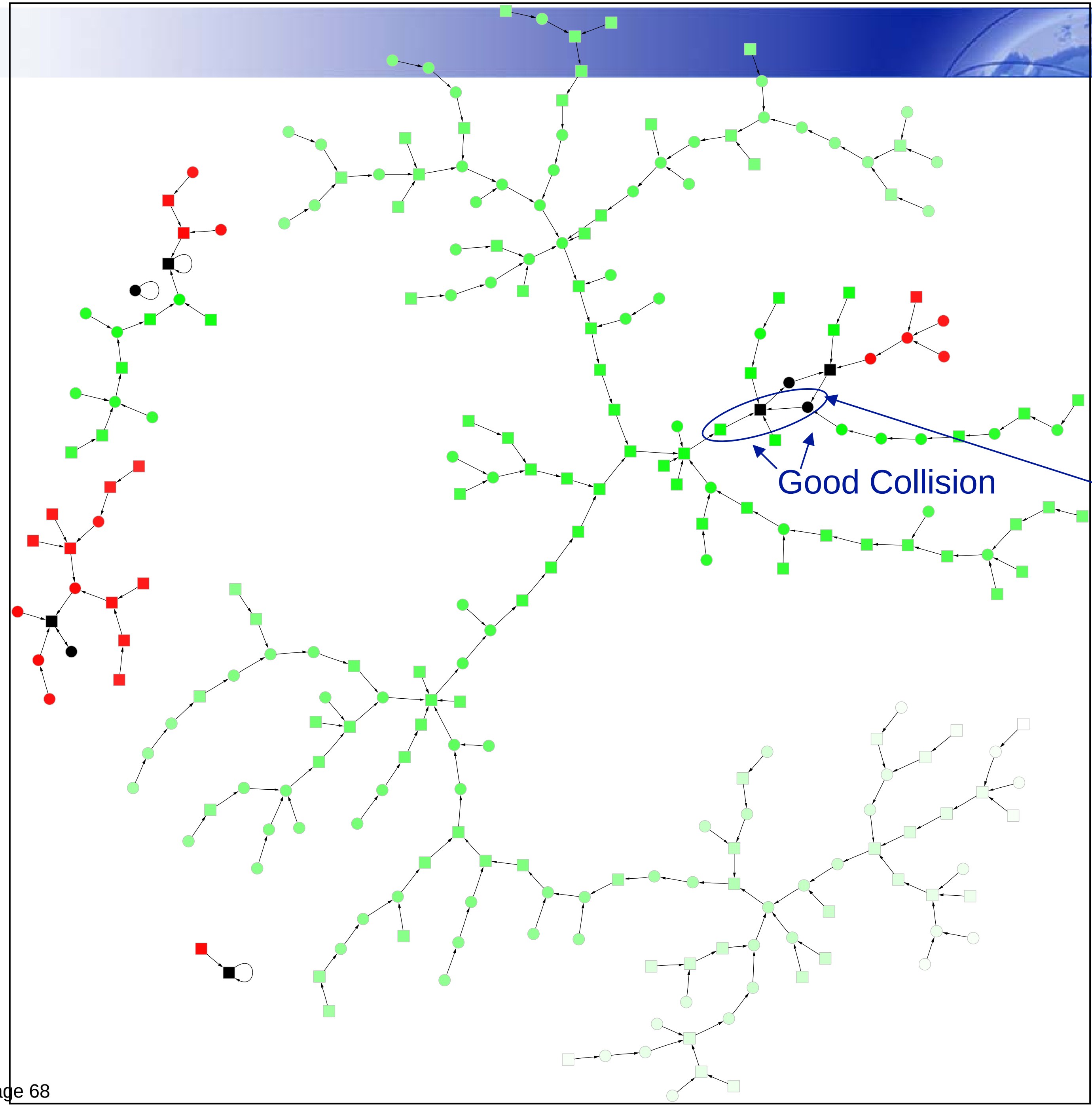
- Visual and interactive presentation of the Floyd algorithm („Moving through the mapping" into a cycle).
- Adaptation of the Floyd algorithm for a digital signature attack.

Starting point

Good collision

Bad collision

* The Floyd algorithm is implemented, but the visualization of the algorithm is not yet implemented in CrypTool.



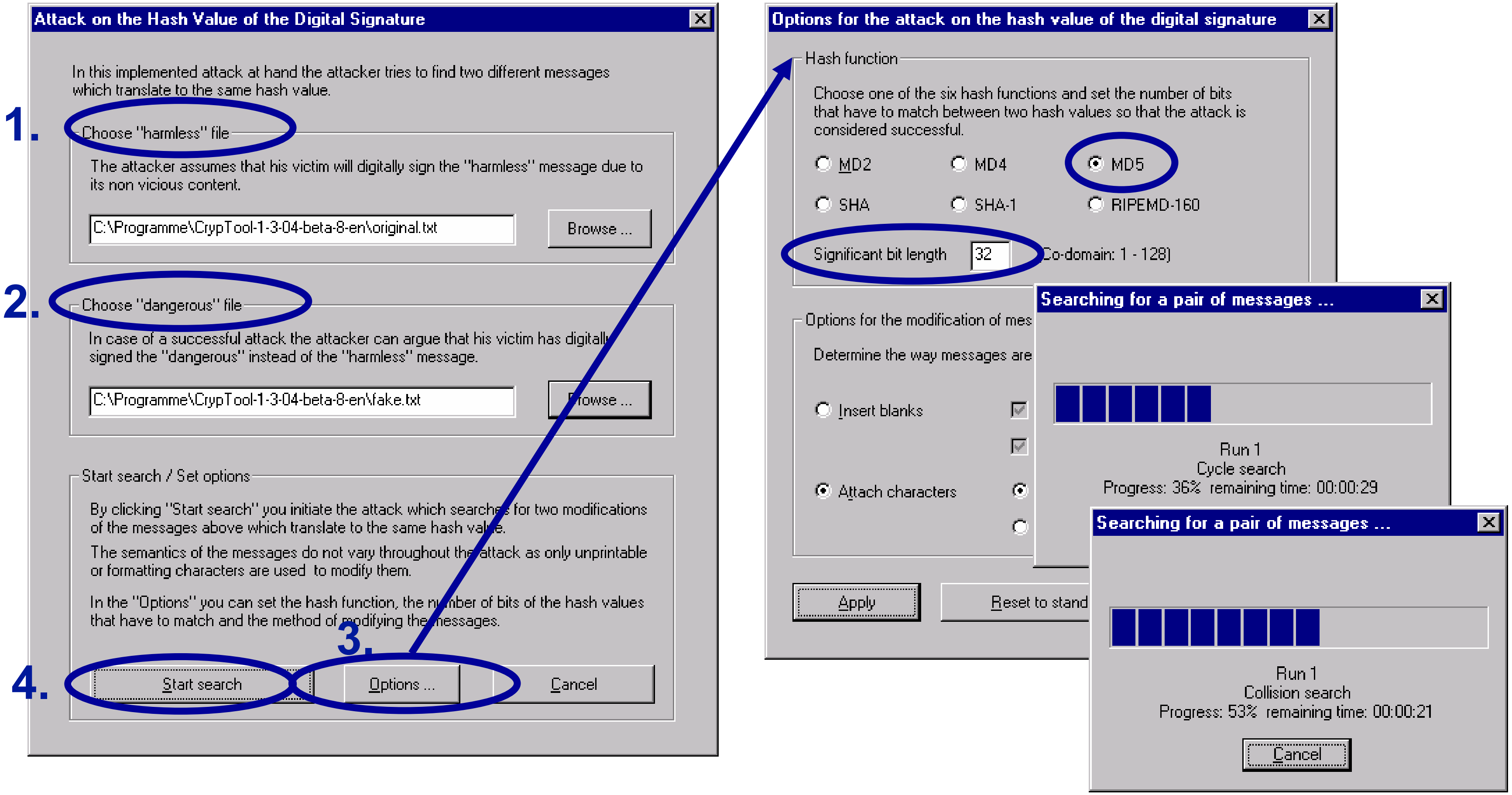
An example for a **“good” Mapping** (nearly all nodes are green).

In this graph almost all nodes belong to a big tree, which leads into the cycle with an even hash value and where the entry point predecessor within the cycle is odd.

That means that the attacker finds a useful collision for nearly all starting points.

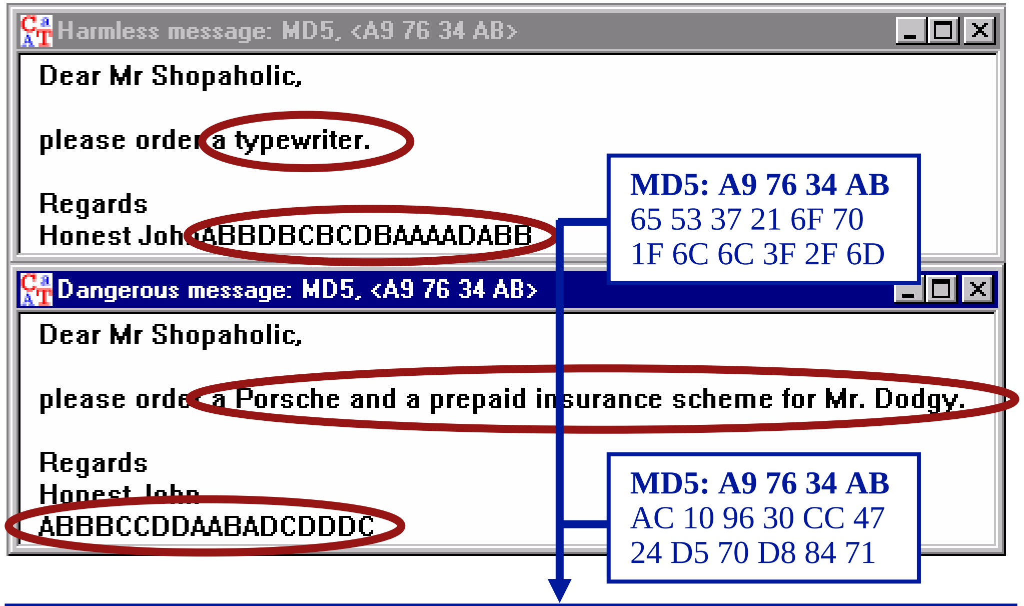
Examples (VII)

Attack on digital signature: Attack



Examples (VII)

Attack on digital signature: Results



The first 32 bits of the hash values are identical.

Experimental results

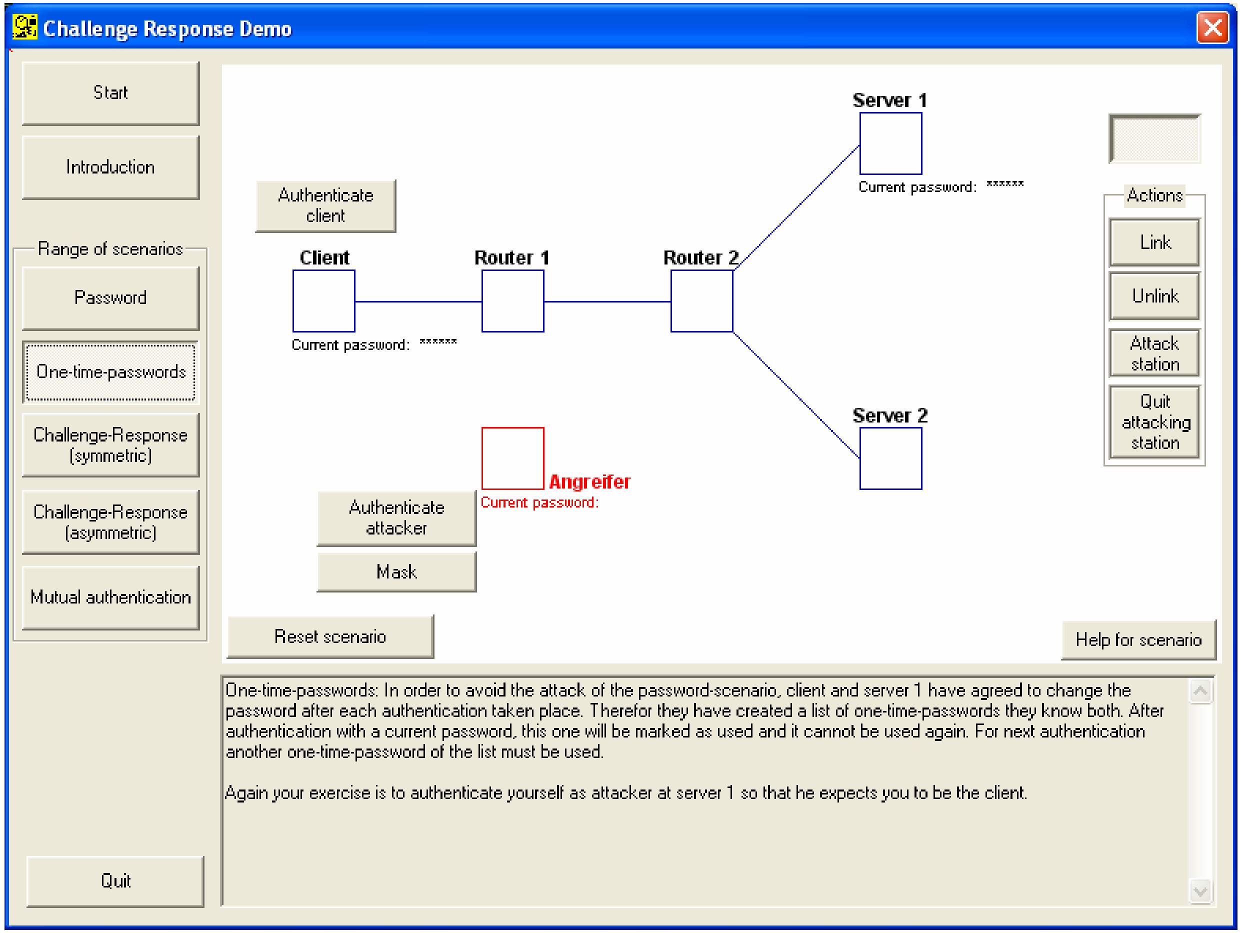
- 72 Bit *partial collision* (equality of the first 72 hash value bits) were found in a couple of days on a single PC.
- Signatures using hash values of up to 128 bit can be attacked today using massive parallel search!

In addition to the interactive handling:

Automated offline feature in CrypTool: Execute and log the results for entire sets of parameter configurations. Available through command line execution of CrypTool.

Examples (VIII)

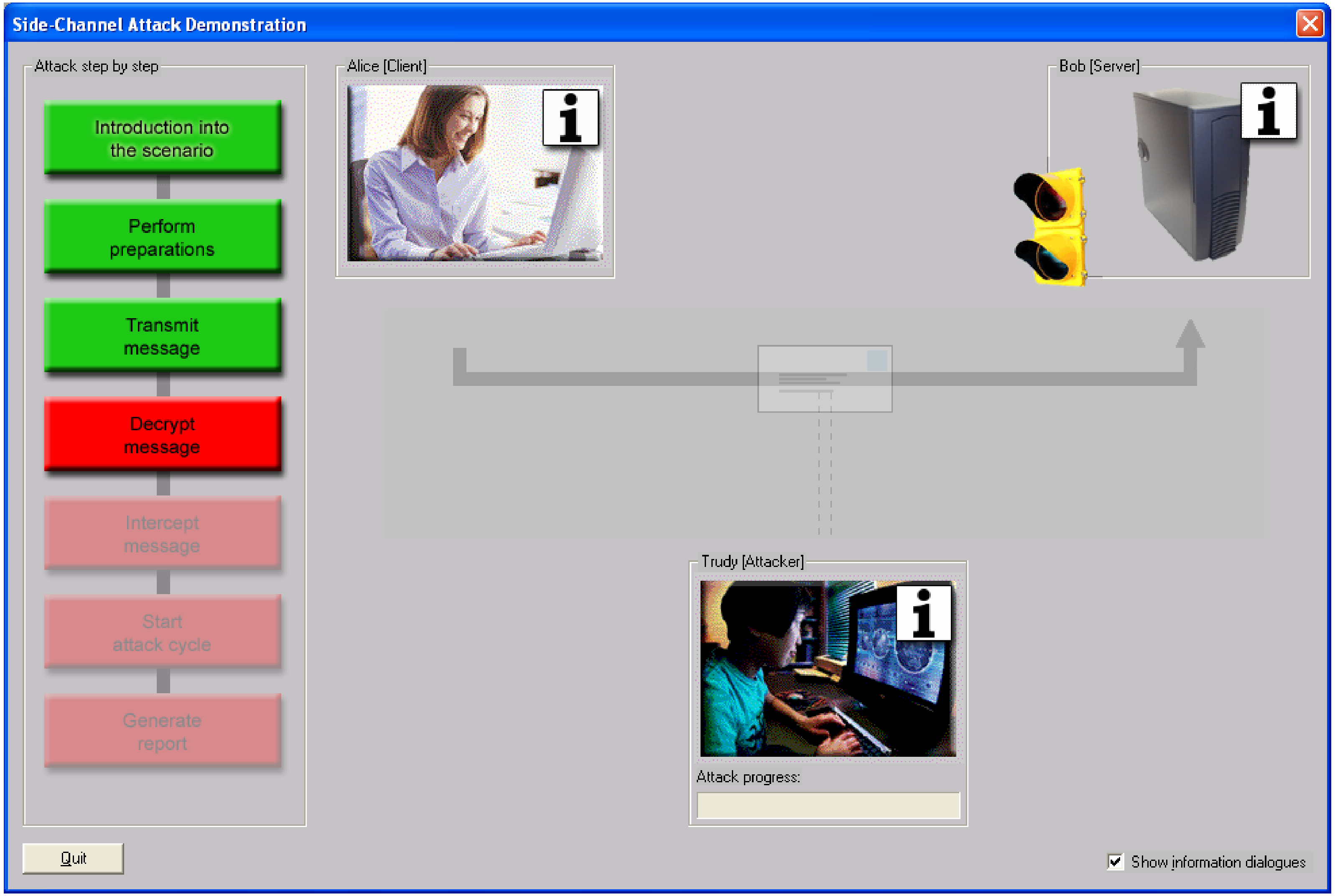
Authentication in a client server environment



- Interactive demo for different authentication methods.
- Defined opportunities of the attacker.
- You can play the role of an attacker.
- **Learning effect:** Only two-sided authentication is secure.

Examples (IX)

Demonstration of a side-channel attack (at a hybrid encryption protocol)



Examples (IX)

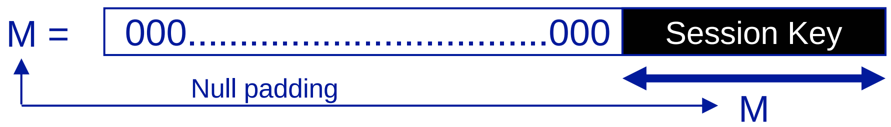
Idea for this side channel attack

- Ulrich Kühn, *Side-channel attacks on textbook RSA and ElGamal encryption* (2003)

Prerequisites:

- RSA encryption: $C = M^e \pmod N$ and decryption: $M = C^d \bmod N$.
- 128-Bit session keys (in M) are „word book encoded“ (Null padding).
- The server knows the secret key d and
  - **uses after decryption the 128 least significant bits only (no validation of Zero padding bits) (that means the server does not recognize if there is something other than zero).**
  - **Prompts an error message, if the encryption attempt results in a wrong session key (decrypted text can not be interpreted by the server). In all other cases there will be no message.**

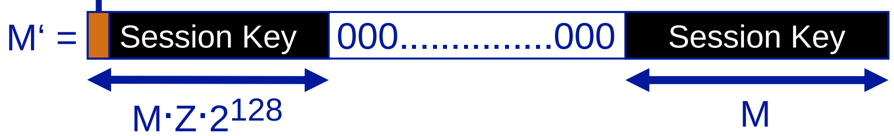
Idea for attack: approximation for Z out of the equation $N = M * Z$ via $M = \lfloor |N/Z| \rfloor$



$C = M^e \pmod N$

All bit positions for Z are successively calculated: for every step one gets 1 bit more. The attacker modifies C to C‘ (see below). If a bit overflow occurs while calculating M‘ on the server (recipient), the server sends an error message. Based on this information the attacker gets a bit for Z.

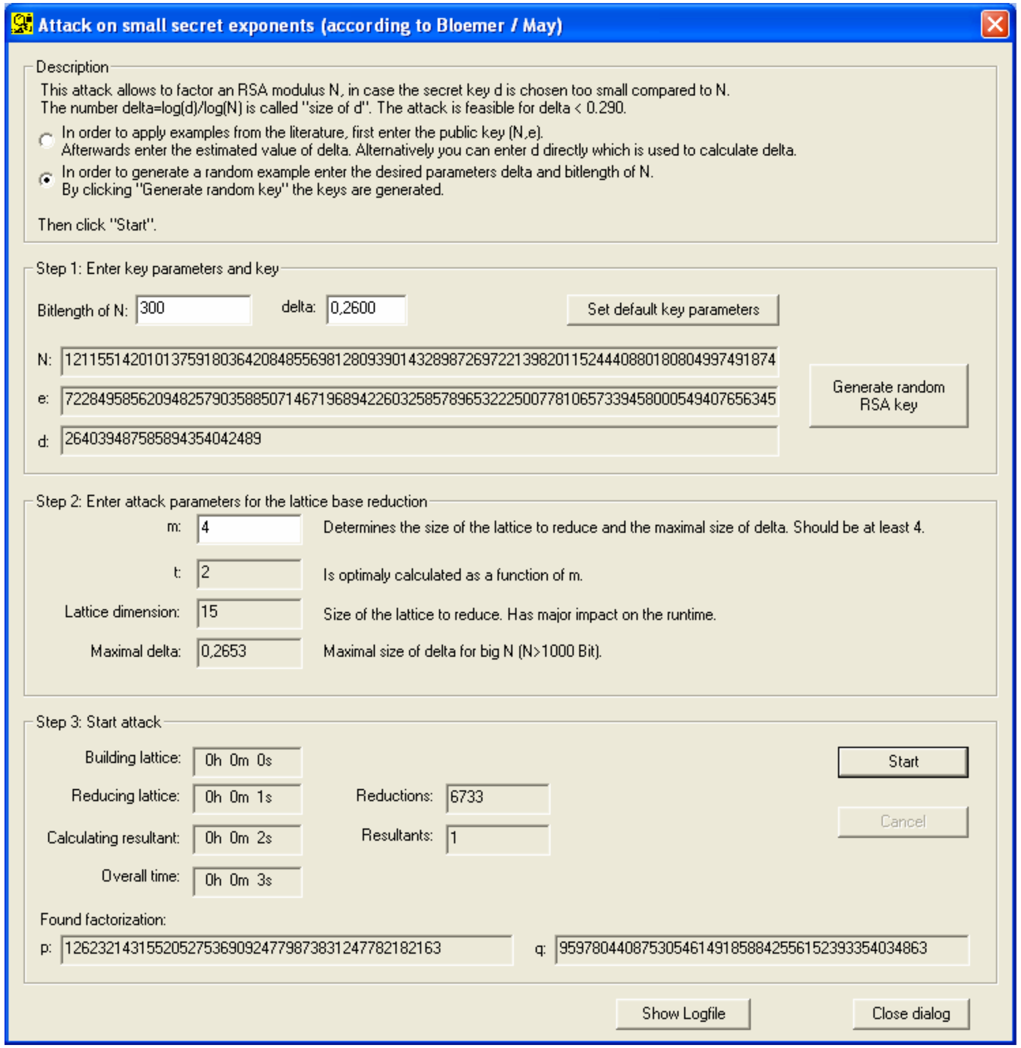
if the most significant bit of M equals ‘1‘, then M‘ unequal M mod 2^{128},



$C' = M'^e = M^e \cdot (1 + Z \cdot 2^{128})^e \pmod N$

Example (X)

Mathematics: Attacks on RSA using lattice reduction



- Shows how the parameters of the RSA method have to be chosen, so that the algorithm resists the lattice reduction attacks described in current literature.
- **3 variants**
  1. The secret exponent d is too small in comparison to N.
  2. One of the factors of N is partially known.
  3. A part of the plaintext is known.
  - These assumptions are realistic.
- The current estimation is displayed.

Examples (X)

Example page of the online help

Help for CrypTool 1.4.00 beta 10

Ausblenden | Zurück | Vorwärts | Abbrechen | Aktualisieren | Startseite | Drucken | Optionen

Inhalt | Index | Suchen

- Start with CrypTool
  - Starting page: An Introductio
  - How to work through the exa
  - How to use the CrypTool onli
- Functionality of CrypTool
- Work on a document
- Change CrypTool options
- CrypTool Menus
  - Menu File
  - Menu Edit
  - Menu View
  - Menu Crypt/Decrypt
  - Menu Digital Signatures/PKI
  - Menu Individual Procedures
  - Menu Analysis
  - Menu Options
  - Menu Window
  - Menu Help

Commands of the menu Lattice Based Attacks on RSA (Menu Individual Procedures \ RSA Cryptosystem)

The menu **Lattice Based Attacks on RSA** contains the following commands:

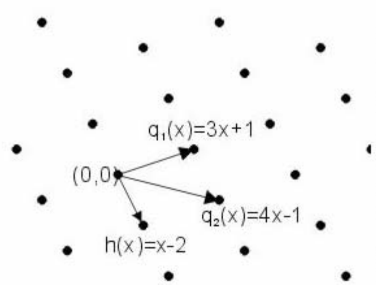
Factoring with a Hint	Attacks RSA with lattice reduction algorthms, if a part of one of the primes of N is known.
Attack on Stereotyped Messages	Attacks RSA with lattice reduction algorthms, if a part of the original cleartext of an intercepted ciphertext is known and if e is small.
Attack on Small Secret Keys	Attacks RSA with lattice reduction algorithms, if d is too small compared to N.

All attacks presented here are based on a common approach: first the task of breaking RSA is transformed into finding the root of a polynomial modulo an integer (mostly $\mathbf{N}$) but to find such a root is a difficult problem.

To solve this problem further polynomials are generated which are known to have the same root. From the coefficients of these polynomials a latticebase is built. This is then reduced with, i.e. the LLL-algorithm to find a small vector.

From this newly found short vector a new polynomial is built. It can be proven that if the vector is short enough, the polynomial has the desired root not only modulo $\mathbf{N}$, but also over the integers.

Example:



The polynomial $q_1(x) = 3x+1$ has a root x_0 **modulo** 7. It is supposed, that the polynomial $q_2(x) = 4x-1$ has the same root x_0 **modulo** 7. From these polynomials the vectors $b_1=[3\ 1]$ and $b_2=[4\ -1]$ are built. All integer linear combinations of these vectors form points in a lattice. The Figure on the left shows a part of this lattice. Each point of the lattice now can again be interpreted as a polynomial having the desired root. A short vector of the lattice is $b_3=[1\ -2]$ from which the polynomial $h(x) = x-2$ is built. this polynomial has a root in $x_0=2$ over the integers as well als **modulo** 7. That $x_0=2$ is also a root of the polynomials $q_1(x)$ and $q_2(x)$ **modulo** 7 can be easily established.
$(3x_0+1=7,\ 7$ **modulo** $7 = 0)$

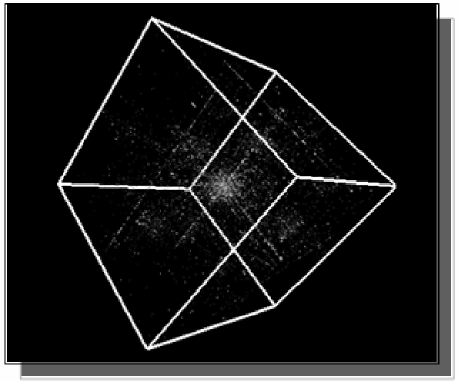
Annotations:

Examples (XI)
Random analysis with 3-D visualisation

3-D visualisation for random analysis

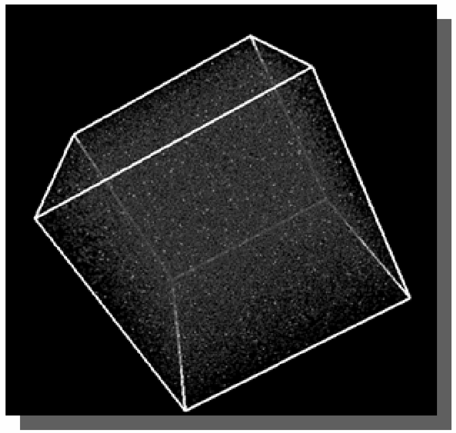
Example 1

- Open an arbitrary file (e.g. report in Word or PowerPoint presentation)
- It is recommended to select a file with at least 100 kb
- 3-D analysis using „Analysis“ \ „Analyse Randomness “ \ „3-D Visualization…“
- Result: structures are recognisable



Example 2

- Generation of random numbers („Indiv. Procedures“ \ „Generate Random Numbers…“)
- It is recommended to generate at least 100.000 random bytes
- 3-D analysis using „Analysis“ \ „Analyse Randomness“ / „3-D Visualization…“
- Result: uniform distribution (no structures are recognisable)

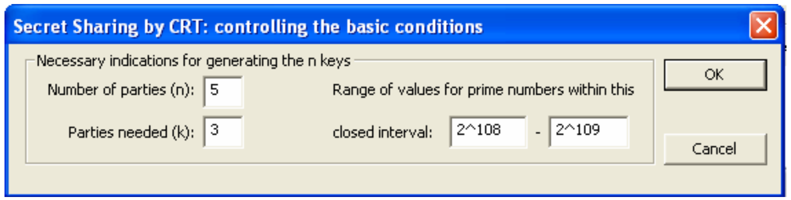


Examples (XII)

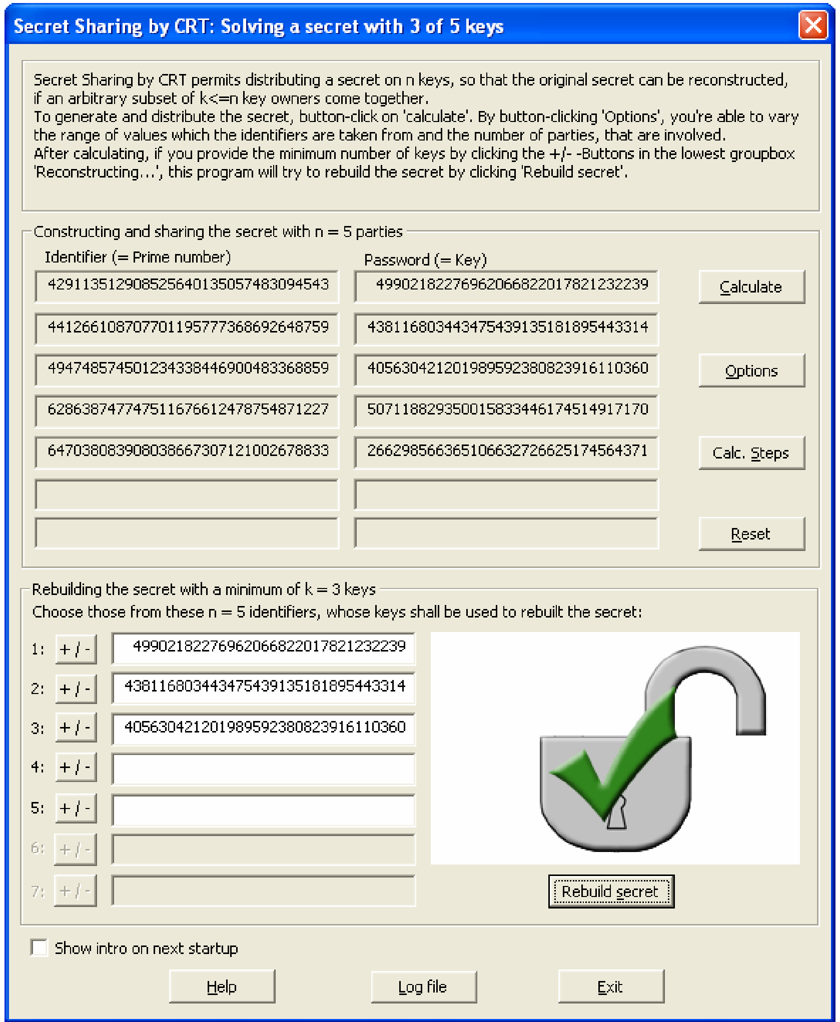
Secret Sharing with CRT – Implementation of the Chinese Remainder Theorem

Secret Sharing Example (I):

- Problem:
  - 5 people get a single key
  - To gain access at least 3 of the 5 people have to be present
- CrypTool: Menu „Indiv. Procedures“ \ „Chinese Remainder Theorem Applications“ \ „Secret Sharing by CRT“
- „Options“ allows to configure more details of the method.



- „Calc. Steps“ shows all steps to generate the key.

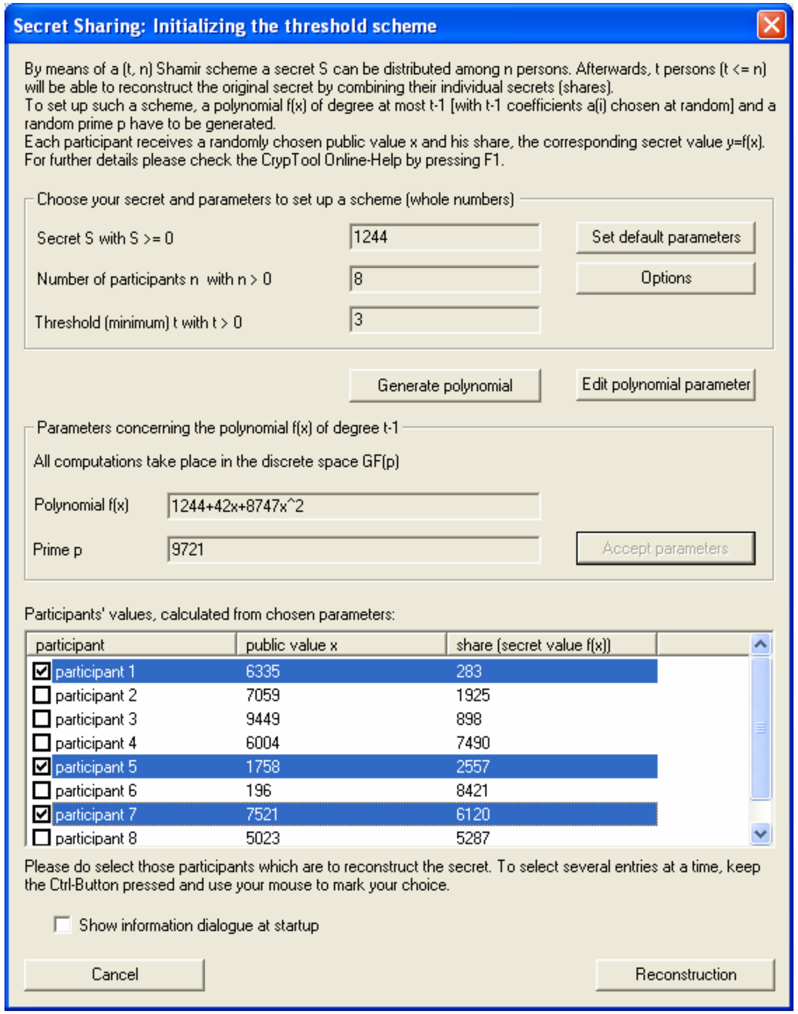


Examples (XII)

Shamir Secret Sharing

Secret Sharing Example (II):

- Problem
  - A secret value should be split for n people.
  - t out of n people are required to restore the secret value K.
  - (t, n) threshold scheme
- CrypTool: Menu „Indiv. Procedures“ \ „Secret Sharing Demo…“
  1. Enter the secret K, number of persons n and threshold t
  2. Generate polynom
  3. Use parameters
- Using „Reconstruction“ the secret can be restored

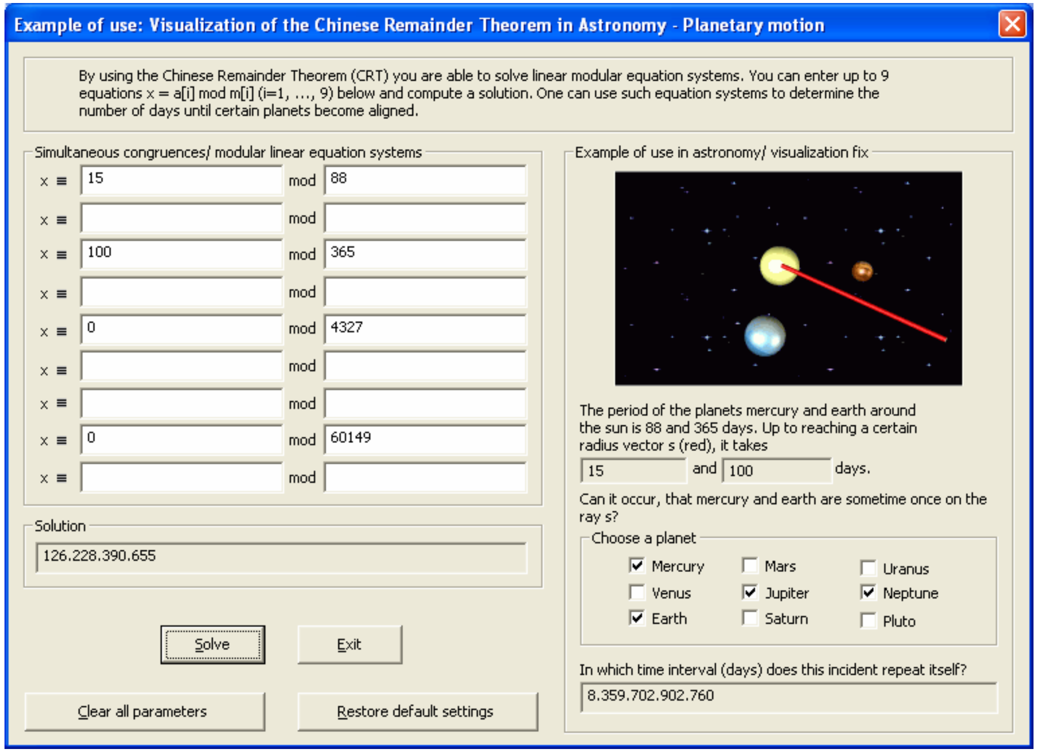


Example (XIII)

Implementation of CRT in astronomy to solve linear modular equation systems

Scenario in astronomy

- How long does it take until a given number of planets (with different rotation times) to become aligned.
- The result is a linear modular equation system, that can be solved with the Chinese remainder theorem (CRT).
- In this demo you can enter up to 9 equations and compute a solution using the CRT.



Examples (XIV)

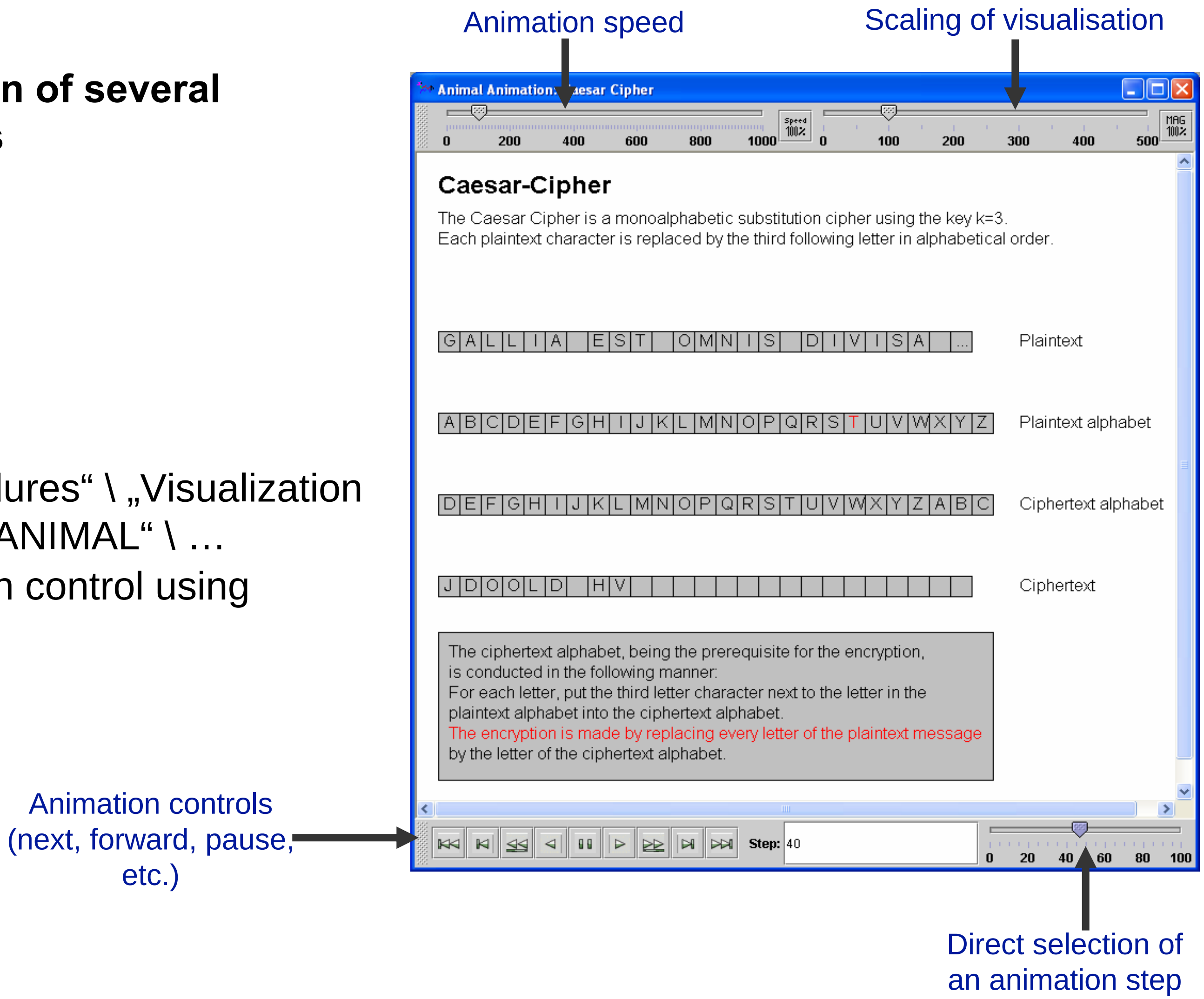
Visualisation of symmetric encryption methods using ANIMAL (1)

Animated visualisation of several symmetric algorithms

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

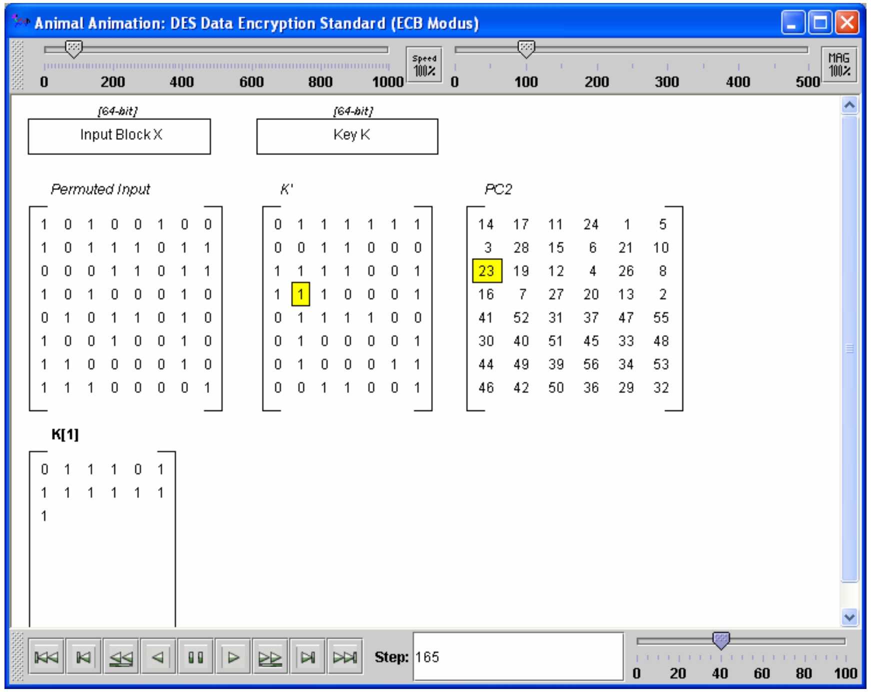
- Menu „Indiv. Procedures“ \ „Visualization of algorithms using ANIMAL“ \ …
- Interactive animation control using integrated control

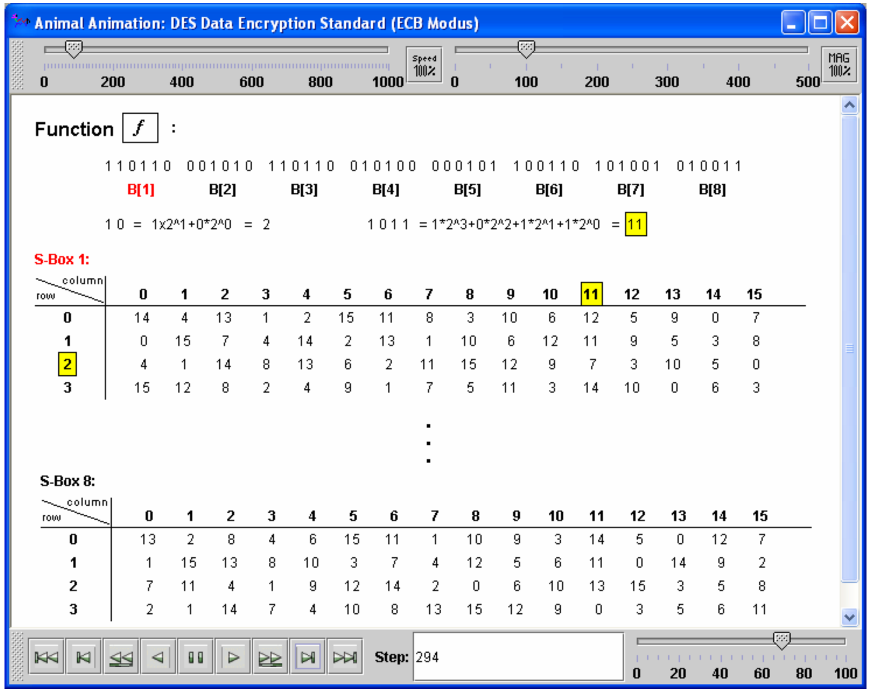


Examples (XIV)

Visualisation of symmetric encryption methods using ANIMAL (2)

- **Visualization of DES encryption**





Examples (XV)

Generation of a message authentication code

Message Authentication Code (MAC)

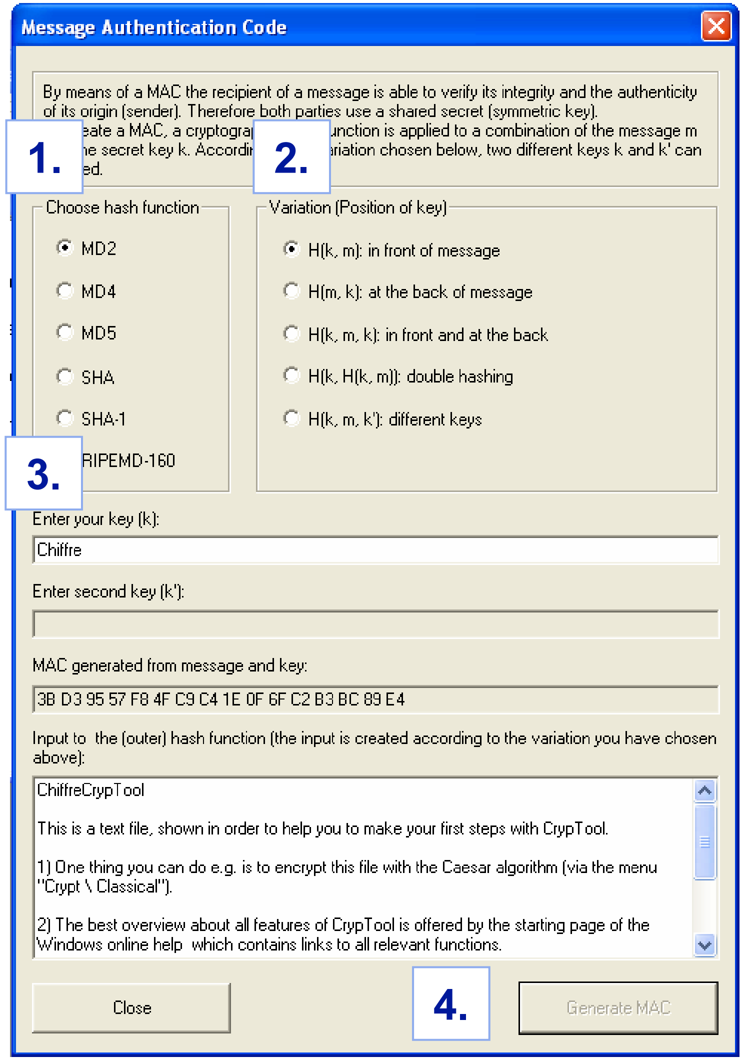
- Ensures integrity of a message
- Authentication of the message
- Basis: a common key

CrypTool

- Menu „Indiv. Procedures“ / „Hash“ / „Generation of MACs…“

Generation of a MAC in CrypTool

1. Choose a hash function
2. Select MAC variant
3. Enter a key (depending on MAC variant also two keys)
4. Generation of the MAC



Examples (XVI)
Hash Demo

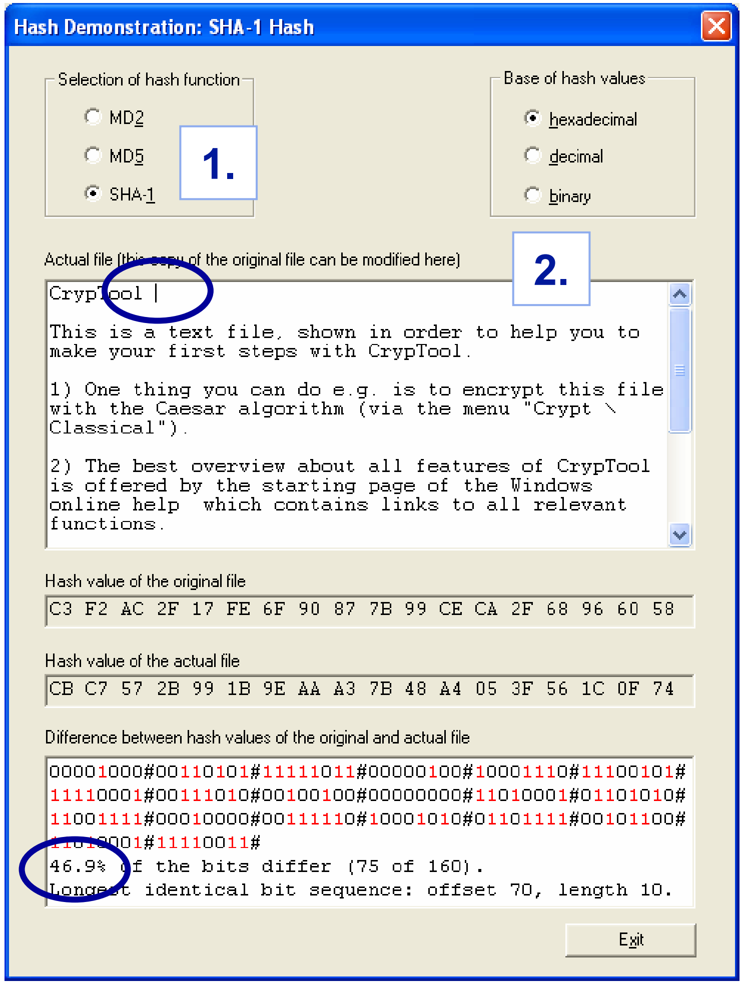
Sensitivity of hash functions to plaintext modifications

1. Select a hash function
2. Modification of characters in plaintext

Example:

Entering a blank after „CrypTool“ in the example text results in a 46,9% change of the bits of the generated hash value.

A good hash function should react sensitive to even the smallest change within the plaintext – *„Avalanche effect“* (small change, big impact).



Content

CrypTool and Cryptography – Overview

CrypTool features

Examples

Project / Outlook / Contact

Future CrypTool development

Planned after release 1.4.00 (see readme file)

- Mass pattern search
- ECC-Demo, ECC-AES hybrid encryption, …
- Visualisation of S/MIME and OpenPGP
- Re-design of CrypTool in Java
- Port of C++ version to WPF / Vista
- Integration of crypto library crypto++ from Wei Dai

Future plans (see readme file)

- Visualisation of protocols (e.g. Kerberos)
- Visualisation of attacks on these protocols
- Visualisation of the SSL protocol
- Demonstration of visual cryptography
- Command line interface for batch processing
- Additional parameters for existing methods / algorithms
- Port of C++ version to Linux

CrypTool as a framework for own developments

Proposal

- Re-use the comprehensive set of algorithms, included libraries and interface elements as foundation
- Free of charge training in Frankfurt, how to start with CrypTool development
- Advantage: Own code does not „disappear", but will be maintained

Current development environment: Microsoft VC++ , Perl,
Subversion source code management

- Until CrypTool 1.3.05: Visual C++ 6.0 only (was available within books for free)
- CrypTool 1.4.00: Visual C++ .net (= VC++ 7.1)(= Visual Studio 2003)
- Description for developers: see readme-source.txt
- Download: Sources and binaries of releases.
  To get sources of current betas, please send an eMail.

Future development environment

- For versions after 1.4.00:
  - C++ version: .NET (without MFC) with Visual Studio 2005 Express Edition (free), WPF and Perl
  - Java version: with Eclipse 3.1, SWT (free)
- Considered:
  - C++ version for Linux with Qt 4.x, GCC 4.0 and Perl

CrypTool – request for contribution

- **Every contribution to the project is highly appreciated**
  - Feedback, criticism, suggestions and ideas
  - Integration of additional algorithms, protocols, analysis (consistency and completeness)
  - Development assistance (programming, layout, translation, test)
    - For the C/C++ project as well as for the new Java project !
    - Especially University faculties using CrypTool for educational purposes are invited to contribute to the further development of CrypTool.
  - Significant contributions can be referenced by name (in help, readme, about dialog and on the CrypTool web site).
  - Currently CrypTool is being downloaded more than 2000 times a month (with 1/3 for the English version).

Contact

Bernhard Esslinger

University of Siegen
University lecturer, Faculty 5, Economics and Business Computing

Deutsche Bank AG
Director, IT Security Manager

esslinger@fb5.uni-siegen.de

www.cryptool.de

www.cryptool.org

www.cryptool.com

additional contacts: See readme within the CrypTool folder

mailing list: cryptool-list@sec.informatik.tu-darmstadt.de

Additional Literature

- Simon Singh, *“The Codebook”*, 1999, Doubleday [English]
- Simon Singh, *“Geheime Botschaften”*, 2000, Hanser [German]
- U. Ulfkotte, *“Wirtschaftsspionage“*, 2001, Goldmann [German]
- Claudia Eckert, *“IT-Sicherheit”*, 3rd edition, 2004, Oldenbourg [German]
- A. Beutelspacher / J. Schwenk / K.-D. Wolfenstetter, *“Moderne Verfahren der Kryptographie”*, 5th edition, 2004, Vieweg [German]
- [HAC] Menezes, van Oorschot, Vanstone, *“Handbook of Applied Cryptography”*, 1996, CRC Press
- Van Oorschot, Wiener, *“Parallel Collision Search with Application to Hash Functions and Discrete Logarithms”*, 1994
- Additional cryptography literature
  (e.g. by Wätjen, Buchmann, Salomaa, Brands, Schneier, Shoup, …)
- Importance of cryptography in the broader context of IT security and risk management
  - See e.g. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder, “Wirtschaftsinformatik”, 2005, Pearson, chapter 14 [German]
  - See Wikipedia (http://en.wikipedia.org/wiki/Risk_management)